NBSIR 88-3692

THE REAL-TIME CONTROL SYSTEM OF THE HORIZONTAL WORKSTATION ROBOT

December 16, 1987

By:
Albert J. Wavering
John C. Fiala

AMRF

U.S. DEPARTMENT OF COMMERCE National Bureau of Standards Gaithersburg, Maryland

THE REAL-TIME CONTROL SYSTEM OF THE HORIZONTAL WORKSTATION ROBOT

Albert J. Wavering
John C. Fiala

December 16, 1987

Certain commercial equipment is identified in this paper to adequately describe the systems under development. Such identification does not imply recommendation or endorsement by the National Bureau of Standards, nor does it imply that the equipment is necessarily the best available for the purpose.

TABLE OF CONTENTS

Page

LIST OF FIGURES

Page

LIST OF TABLES

THE RCS-II/T3 CONTROL SYSTEM

I. INTRODUCTION

This chapter gives a brief description of the Real-time Control System II (RCS-II) developed by the National Bureau of Standards (NBS) and describes how this manual is organized. A short discussion of the relationship between RCS-II other versions of RCS is included. The chapter also describes the documentation conventions used throughout the manual to indicate different types of information. The conventions include how the manual presents syntax descriptions of the languages within RCS-II. Finally, the chapter describes who should read this manual.

1. WHAT IS RCS-II?

RCS-II is a microprocessor-based system for the real-time control of automated systems. The National Bureau of Standards (NBS) has been developing such systems for ten years. The results of this research are embodied in two closely-related versions of RCS, the RCS described in The NBS Real-time Control System User's Reference Manual [1] and RCS-II, presented here. A significant difference between the systems is that RCS-II does not support master/slave operation of a number of processors from a single terminal, background tasks, and interrupt routines, whereas RCS does. Otherwise, RCS and RCS-II are functionally very similar. Reference [1] will therefore be frequently referred to. The majority of the current manual, however, is concerned with the application of RCS-II to the control of a machine tool-tending T3 robot in the Automated Manufacturing Research Facility (AMRF) [2]. This application, which will be referred to as RCS-II/T3, is not covered in detail in [1].

2. ABOUT THIS MANUAL

This section introduces the organization of the manual and the documentation conventions used.

2.1. How This Manual is Organized

This manual is organized into 9 chapters, 2 appendices, and a list of references. This chapter, "Introduction", introduces RCS-II and explains how to use this manual.

Chapter 2, "System Overview", provides a brief description of the role of the T3 robot in the Horizontal Workstation (HWS) of the AMRF, and the RCS-II/T3 architecture.

Chapter 3, "Electrical Hardware Configuration", identifies and describes the computer, sensor, and other electronic hardware used for this robot application.

Chapter 4, "Using RCS-II/T3", provides step-by-step procedures for several RCS-II operations and utilities. It explains how to startup and shut down the various systems, and how to run the robot with RCS-II/T3.

Chapter 5, "Primary Control System Processes", contains information on the four primary processes which control the motion of the robot; the Prim(itive) process, the E(lemental)-move process, the Subtask process, and the Task process. The function, commands, and interfaces of each process are discussed.

Chapter 6, "Other Control System Processes", provides the same type of information for other processes in the system. These include the processes that interface RCS-II/T3 with the robot, control gripper and quick change operation, and perform diagnostics and graphics functions.

Chapter 7, "Task Data", describes the data structures used to define robot motions and associate them with particular objects.

Chapter 8, "Entering Data for a New Part: Example", provides an example of how all the data needed to make a particular part is determined and entered into the system.

Chapter 9, "External Communications", describes the nature of RCS-II/T3 communications with external systems such as the HWS and the database.

2.2. Documentation Conventions

This manual uses the same documentation conventions as ref [1], repeated here for convenience:

- The name of each keyboard key mentioned in the text appears in uppercase letters. For example, the carriage-return key appears as RETURN.

- In a control sequence, a caret (^) represents the CONTROL key. For example, control-C appears as ^C.

- Information that you must enter exactly as it appears in the manual is underlined. For example, "Enter 3 LOAD" means to type 3 LOAD and press RETURN.

- Variable data that you are to enter is represented within

square brackets ([ ]). The instruction "Enter [block#] LOAD" indicates that you are to replace [block#] with a specific block number when you enter the LOAD command. For example, you might type 3 LOAD and press RETURN.

- The terms "position", "point", and "force" will imply a generalized interpretation (i.e. position/orientation and force/torque) unless the context indicates otherwise.

3. WHO SHOULD USE THIS MANUAL

This document is intended for NBS personnel and for government, industry, and university researchers interested in robot control system implementations within the Automated Manufacturing Research Facility (AMRF). The manual assumes that you are conducting research and developing applications for the real-time control of robots using sensors, hierarchical task decomposition, and multiple CPU's. Although the manual includes basic instructions for operating RCS-II and running the T3 robot, substantial training beyond the scope of this document would be required for safe operation of the system.

This manual assumes that you are familiar with the concepts and terminology of robotics and with the FORTH programming language used extensively in RCS-II. The polyFORTH 1 Reference Manual and the polyFORTH 8086 Operations Manual from FORTH, Inc. of Hermosa Beach, California provide in depth information on the FORTH language and operating system used for RCS-II. In addition, you are encouraged to read reference [1] (see Appendix A) to become familiar with RCS, keeping in mind the differences mentioned in the first section. Most RCS-II/T3 code is written in a language called SMACRO, which consists of macros written in FORTH. Chapter 7 of reference [1] discusses the details of SMACRO, and should be read before using RCS-II/T3.

II. SYSTEM OVERVIEW

This chapter gives an overview of the RCS-II/T3 Control System, describing its role in the Horizontal Workstation and explaining the basic concepts involved in its operation. The discussion of RCS-II basic concepts, although of limited detail, covers ideas of fundamental importance to all subsequent discussion in this document. For this reason, a thorough reading of section 2 of this chapter is highly recommended. But first, section 1 reviews the tasks performed by RCS-II/T3 for the Horizontal Workstation.

1. OVERVIEW OF THE T3 ROLE IN THE HWS

The Horizontal Workstation (HWS) produces machined parts on its horizontal machining center. The parts enter the workstation either as unfinished parts from other workstations, or as blanks from stock. These parts enter and exit by way of the material buffering device [3]. The material buffer accepts and delivers part trays to the material handling system, as well as presenting trays to the Cincinnati Milacron T3 manipulator, which tends the horizontal machining center. Figure II.1 shows location of the various pieces of equipment within the workstation.

In addition to controlling the T3 robot, the control system is responsible for controlling the Active Pedestal for regripping parts. An overview of the Pedestal subsystem, along with the other subsystems related to tending the horizontal machining center, is presented below.

1.1. Tasks Performed by the T3

The role of the T3 in the Horizontal Workstation is to tend the horizontal machining center. This means that the robot must take care of all manipulation tasks related to making parts that are not otherwise automated. Primarily, this means putting parts into and taking parts out of the machining fixtures. In addition, the T3 must change tools on the horizontal mill.

A typical machining cycle finds the material buffer presenting a tray with a part blank to the T3. As instructed by the Horizontal Workstation Controller (HWSC) [4], the robot takes the part from the tray and inserts it into a fixture [5]. After the fixture closes on the part, the robot often can simply release it and move to a safe position. In some instances the robot must make sure the part is seated properly. In this case, the robot releases the part, moves to a position in front of it, waits for the fixture to open, and then presses the part against the back of the fixture jaws. When the fixture closes again the part should be properly seated.

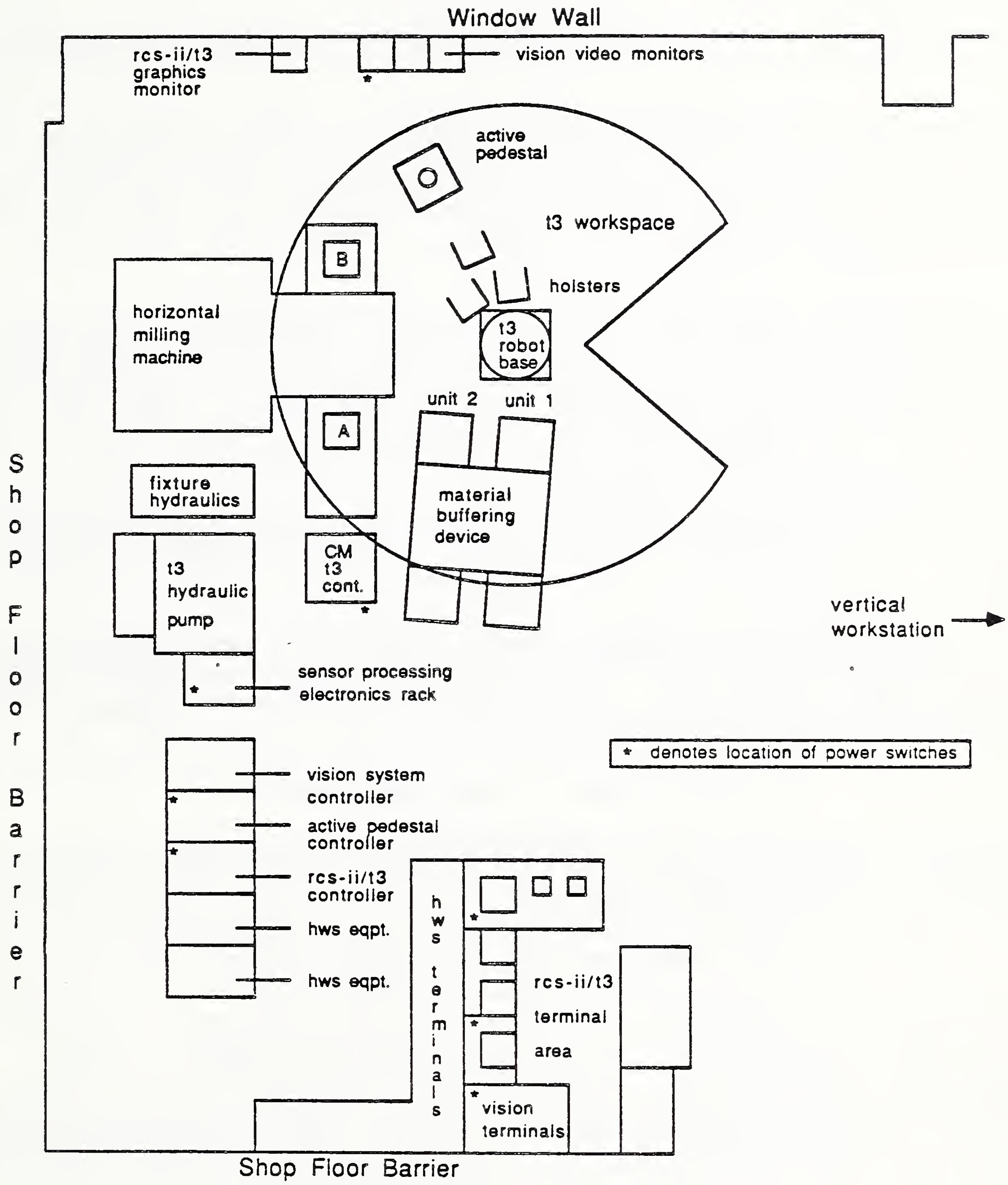


Figure II.1. Horizontal Workstation Layout

When the machining of the part is complete, the RCS-II/T3 is commanded by HWSC to acquire the part in the fixture. Once the robot has the part, the fixture opens and the T3 is told to place the part in a tray presented by the material buffering device.

Sometimes, a part must be refixtured in order to complete the machining operations, e.g. to machine both sides of a part. When this must be done, the RCS-II/T3 Control System uses the Active Pedestal to turn the part around and regrip it. This allows the robot to put the part back into the fixture in a different orientation.

Another major operation of the T3 is to change tools on the horizontal machining center. The tool drum of the machining center can hold but a limited number of tools. When new tools are needed, either to make new parts or because old tools are worn out, the robot can take old tools out and put new ones in. Typically, the HWSC tells RCS-II/T3 to get a tool from the tool drum and put it in an empty sector of a tool tray presented by the material buffer. Then the robot is instructed to get a new tool from a tray and place it into the empty location in the tool drum.

One final operation carried out by the T3 is rather specialized. On rare occasions the robot is told to pick-up a brush and brush chips from the fixture. The brush is stored in a special bracket between the gripper holsters.

1.2. <u>Special-purpose Robot Equipment</u>

To assist in carrying out the above mentioned tasks, the T3 requires certain "special-purpose" equipment. This equipment is controlled by RCS-II/T3.

1.2.1. Grippers and Quick Change

To manipulate different types of objects, the robot requires different end-effectors (or grippers.) For tasks in the Horizontal Workstation, three grippers are needed; one for changing tools, one for manipulating prismatic parts, and one for manipulating cylindrical parts. The T3 can change end-effectors using the Quick Change device attached to its wrist. All of these devices are controlled by the RCS-II/T3 Control System.

The gripper for changing tools is called the "tool gripper". This gripper is a parallel split-rail device which operates by pneumatic cylinder. It is equipped with a linear pot for sensing finger position, and a six-axis force/torque sensor. When not in

use the gripper is stored in the freestanding holster in front of the robot base, henceforth called the "tool gripper holster".

The gripper used for manipulating prismatic parts is called the "parts gripper". This gripper is also used to do brushing. The fingers of the gripper are instrumented to sense grasp force. The finger opening can be servo controlled by controlling valves that damp the flow of air to the pneumatic actuator. This gripper is stored in the righthand (from the point of view of the robot) holster attached to the robot base.

The cylindrical parts gripper is often called the "v-blocks gripper," or "v-gripper", because of the dual v-blocks on the fingers used for grasping round objects. Like the tool gripper, the v-gripper is a pneumatically actuated, split-rail design. The split-rail gripper design is discussed in detail in [6]. Its only sensor is a linear pot for finger position, and it is stored in the lefthand holster attached to the robot base.

The sensor signals and the air for the pneumatic actuators of the grippers are connected to the rest of the robot via the electrical and pneumatic ports of the Quick Change device. The Quick Change consists of two types of interlocking plates, an "A" plate which is permanently attached to the manipulator wrist and "B" plates which are attached to the grippers. The A plate is equipped with a locking pin and an actuator so that a B plate can be locked on and unlocked off the robot. When the plates are mated, then the gripper can be actuated and its sensors monitored through the Quick Change ports. The Quick Change device is more thoroughly described in [7].

During normal operations, RCS-II/T3 will sometimes decide that a different end-effector is needed for the task at hand. When this happens the manipulator is guided through a sequence that results in the current gripper being returned to the proper holster, and the new gripper connected and removed from its holster. This operation is called a "quick change".

1.2.2. Vision System

The vision system is used by the robot to determine the location of part blanks in a tray. It is generally used only with prismatic parts.

When the robot is instructed to get a part from a certain sector of a tray, it may not know completely the position and orientation of the part within the sector. (Note that prismatic parts are not required to be placed in a fixed position or orientation for processing in the Horizontal Workstation.) For

this reason, the control system moves the robot to a position where the camera attached to the robot's wrist is above the sector that contains the part. The RCS-II system then asks the vision system to locate the part. If the vision system can identify the part, it returns the part's position and orientation so that the robot can pick it up.

The reader is referred to references [8,9] for further information on the vision system.

1.2.3. Active Pedestal

The Active Pedestal, as mentioned above, is used to hold parts so the T3 can regrip them. In this sense, the Active Pedestal acts as an second hand for manipulating parts. This capability is important when a part must be turned around or otherwise refixtured.

The Active Pedestal has its own RCS-based control system, a system completely separate from RCS-II/T3. The Active Pedestal controller and RCS-II/T3 system communicate by way of the factory network. This is discussed in IX.3.

The Active Pedestal device consists of a gripper similar to the parts gripper mounted on a table rotated by a stepper motor. The rotary table is fixed on a metal stand. This stand is not bolted to the floor and thus can be moved from its known position. It is very important that the Pedestal stand be positioned where marked on the floor. All operations with the Active Pedestal assume it is in this one, known position.

2. OVERVIEW OF THE RCS-II ARCHITECTURE

This section discusses the structure and features of RCS-II [10,11]. The concepts of hierarchical task decomposition, generic control processes, cyclic execution, and common memory as embodied in RCS-II are explained in detail.

2.1. Hierarchical Task Decomposition

Hierarchical task decomposition is an important technique for organizing the complex information processing required for real-time sensory-interactive robot control. This concept is the basis for the organization of the RCS-II/T3 Control System.

The idea behind hierarchical task decomposition is much the same as in structured programming. The control system is organized into several levels. Each level decomposes the task into simpler subtasks as it passes down the hierarchy (see Figure II.2). Thus

a complex task such as TRANSFER, entering at the top-most level, can be decomposed into drive signals for the robot's actuators, the output of the lowest level. Each level can be written so that it is modular, i.e. having clearly defined interfaces to other levels, and more understandable since it deals with a smaller amount of information.

In RCS-II/T3 the hierarchical task decomposition is realized as shown in Figure II.3. Commands from the Horizontal Workstation Controller are received by the Task Level and broken down into subtasks. These subtasks are commands to the Subtask Level. The Subtask Level decomposes the commands into commands to the Elemental-move (E-move) Level. The E-move commands are further decomposed into commands to Prim. The Prim Level breaks its commands into small trajectory points that are passed through the T3 Interface Level to the T3 Controller.

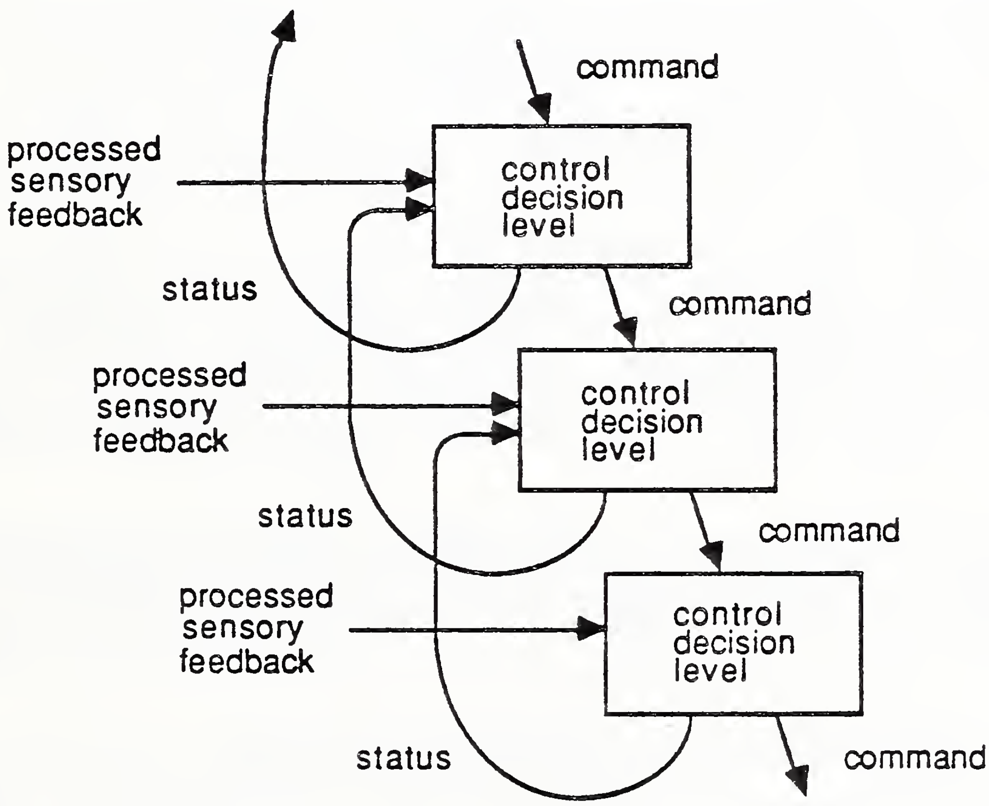


Figure II.2. General Hierarchical Control Structure

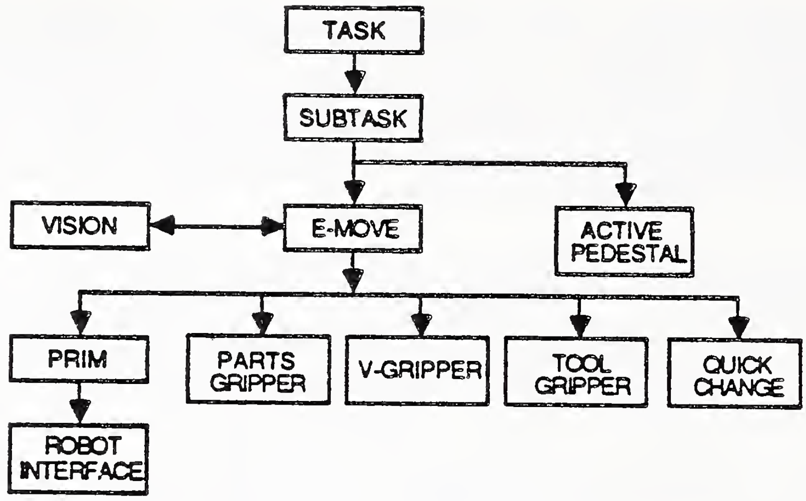


Figure II.3. RCS-II/T3 Control System Hierarchy

Note that E-move also sends commands to the grippers and Quick Change. Thus, E-moves are decomposed not only into Prim commands, but into a combination of Prim, gripper and, occasionally, Quick Change commands. E-move also requests sensory information from the vision system when required.

The Subtask Level has the added responsibility of commanding the Active Pedestal. The Active Pedestal has its own internal decomposition into gripper and table commands. Although Task only commands Subtask in the control hierarchy, it also updates the information in the AMRF Data System. This can be thought of as an update to the AMRF's World Model based on the action just taken by the robot.

As indicated in Figure II.2, each Level returns status back to the Level above. This is true in the RCS-II/T3 hierarchy as well, although these arrows are not explicitly shown in the figure.

2.2. Generic Control Process Structure

Each box of the RCS-II/T3 hierarchy shown in Figure II.3 is implemented as a RCS control process. A control process is a software entity that implements a set of related system functions and communicates with other processes in the system only through explicitly-defined interfaces. Every control process follows a basic generic format. This format is depicted in Figure II.4. The generic process consists of preprocessing, decision processing, and postprocessing as will be detailed in the next three sections.

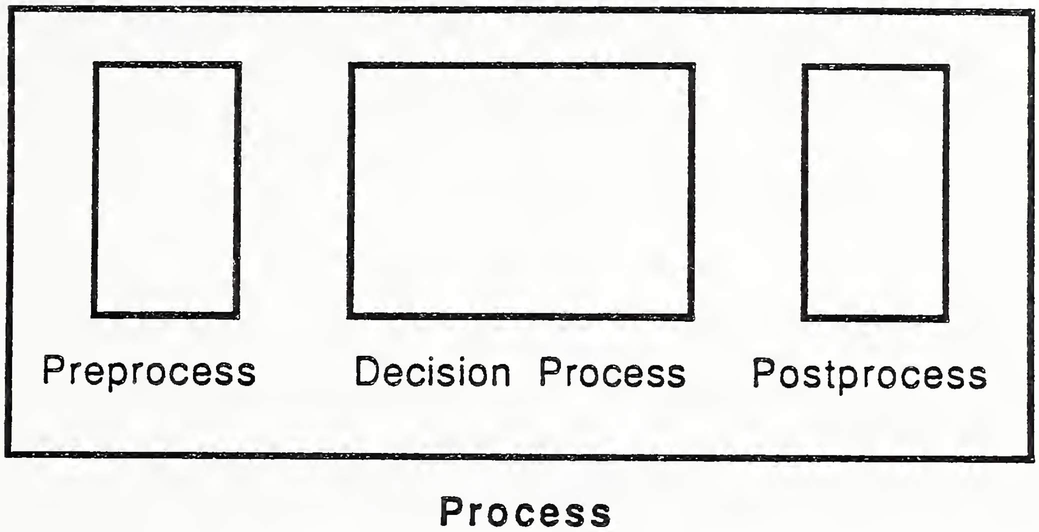


Figure II.4. Generic Control Process

2.2.1. Preprocessing

The preprocessing section of the control process has three major functions. First, all input communication buffers must be filled. Then any sensory data is obtained and processed. And finally, the state variables are set for the decision processes.

The input communication buffers are filled by calling routines that read the common memory areas where the communication data is stored. How this is done will be detailed in Section 2.3 of this chapter. The input data usually consists of statuses from lower-level processes and the command from the higher-level process.

After performing the "read" operation, preprocessing must sample any sensors that directly connect to the process. Some processes have many sensors and some have none. Task, for example, reads no sensors directly--its high-level decomposition does not involve direct sensory feedback. The Prim Level, on the other

hand, reads a number of sensors, including the force sensors during tool changing. This is necessary since Prim must modify its outputs based directly on the values of the sensors. If the raw sensor data requires any processing, this is also done by the preprocessing part of the control process.

Once all data is obtained and processed, preprocessing sets state variables according to the input. The state variables appear explicitly in the state tables of the decision processing section. The next section describes the state tables and their purpose.

2.2.2. Decision Processing

The decision processing part of the control process makes decisions regarding outputs based on inputs. The idea here is to use state tables that can be kept simple enough to be easily understood by the programmer. All complex processing is placed elsewhere so that the state tables only have a few states and can be easily read. This is why preprocessing sets state variables for the state tables rather than allowing the state tables to operate directly on the input itself.

Decision processing is usually composed of a two-level hierarchy of state tables. The top state table is used to call the appropriate state table for the current input command. Once in the current input command state table, the state variables select a certain row of the table. The action specified by this row is taken. The action often involves setting the output command buffer to get the desired behavior for a lower-level process.

2.2.3. Postprocessing

The control process completes a cycle of execution in postprocessing. Postprocessing has two functions, set the output variables based on the decision made, and write the output into common memory.

Setting the output variables may involve some calculations, but often the amount of computation is minimal. The most important part of postprocessing is communicating the outputs. To do this postprocessing writes the output buffers into the designated areas of common memory. The appropriate control processes will pick-up this data when their preprocessing sections read it from common memory.

2.3. Cyclic Execution

The control processes of the system execute cyclicly. That is,

each process executes completely once every system cycle. The RCS-II/T3 system is on a 40 millisecond system cycle. So every 40 milliseconds the processes of the system execute.

Every 40 milliseconds a pulse on the system bus tells the processes that they may begin executing. When a process sees the falling edge of the pulse, it executes once. That is, it does preprocessing, decision processing, and postprocessing. Then the process waits until the next system pulse is received.

The system pulse is generated from board 0. The word that starts board 0 executing, SSGO, also starts sending the pulse out on the bus. If board 0 is not executing, then no process in the system will execute since there is no pulse.

Even when the system pulse is present, not all processes in the system will execute. It is possible for a process to be "dormant". A dormant process is prevented from executing by having its execution flag set to "false". The usefulness of this is in being able to disable processes that are not needed. One example on this in RCS-II/T3 is the gripper processes. There is a control process defined for each gripper; however, the robot can only operate with one gripper on at a time. Thus, the processes for the two grippers not in use are made "dormant" and cannot affect the system.

2.4. Common Memory

Each processor in the system has some local memory that is used only by that processor. However, much of the data in the system is shared among the processors. The sharing of data is accomplished through common memory. This is memory in the address spaces of all the processors, which can be accessed by all processes in the system.

Some of the rationale for having a large common memory must be attributed to the global dictionary required by the FORTH-based operating system. This dictionary takes up a large chunk of common memory. However, as far as system operations go, the main use of common memory is in shared data.

The task data, for telling the robot how and where to move for each object the system knows how to manipulate, is stored in common memory. Different processes access this data based on need. Having this data in common memory also simplifies the information in the interfaces between processes. For example, to tell Prim to move to a certain position, E-move does not have to explicitly pass the position information. E-move can pass a pointer to Prim to indicate where the position information can be

found in common memory.

Common memory in RCS-II/T3 stores shared data by way of "files". These files are sections of memory that contain a sequence of records, each record having a predefined format. The records are usually linked together in a linked-list structure, although this is mostly transparent to the casual programmer. Files are used for communications between processes.

2.4.1. System Dictionary

As mentioned previously, the system dictionary resides in common memory. The dictionary is a FORTH-style dictionary that contains all the words of all the vocabularies defined in the system. Familiarity with FORTH is a requirement to understanding operations related to the system dictionary.

A vocabulary is defined for each control process, so that words and variables defined for each process can be kept somewhat separate. For example, there are separate vocabularies for Task, Subtask, E-move, Prim, etc. Words defined in one vocabulary can not be accessed when working in another vocabulary.

When an operator enters a word in RCS-II/T3, the operating system searches the system dictionary for it. It does not search the whole dictionary, but only those vocabularies currently available to the operator. Since everything is defined through the system dictionary, processes must have access to common memory through the system bus in order to execute. When the bus is not available, the entire system comes to a halt.

Processors do have control process routines in local memory, however. This helps limit the amount of bus activity during process execution. Also, every processor has a local operating system so that it is possible to work with individual boards directly, even without access to the system dictionary.

2.4.2. Communications Buffers

As has been inferred in the above, processes communicate by passing data in common memory buffers. These buffers are defined as files with only one record. The format of the record is the interface between the two processes.

The preprocessing routines read the data from the common memory buffer. The postprocessing routines write data to the common memory buffer. Since these operations occur every system cycle, the processes are continuously reading inputs and writing outputs, and in this manner, communicate.

2.4.3. Task Data

The task data for RCS-II/T3 is all of the task dependent data required to carry-out the system's tasks. This data is shared between processes and is stored exclusively in common memory.

Task data for the system includes position data that defines the locations known to the system, and object data that gives object dependent information. Object dependent information can be position-related, telling the system how and where to move the robot to manipulate the object. It also can tell more general information such as what gripper should be used to grasp the object, and whether the object should be recognizable to the vision system.

Extensive details on task data are given in Chapter VII.

III. ELECTRICAL HARDWARE CONFIGURATION

This chapter will describe the electrical hardware used in the RCS-II/T3 application. The required computer hardware will be identified, and the numerous sensors and their related processing hardware will be described. Schematics of processing electronics and cabling diagrams may be found in Appendix B.

Figure III.1 is a block diagram of the electrical hardware components of the RCS-II/T3 system. The diagram shows the computer hardware, the sensors and sensory processing electronics, and the commercial T3 robot controller. A similar block diagram for the Active Pedestal electronic hardware is shown in Figure III.2. The Active Pedestal computer boards share their own backplane separate from the RCS-II/T3 controller. Communications between the two systems takes place over the network. The analog interface box is shown in both figures, since it serves as a buffer for analog signals associated with both systems.

1. RCS-II/T3 Computer Hardware

The RCS-II/T3 computer hardware consists of:

- o Intel iSBC 86/30 single-board computers
- o Intel iSBX 311 analog input multimodules*
- o Intel iSBX 350 parallel I/O multimodules*
- o Intel iSBX 351 serial I/O multimodules*
- o Intel iSBC 337A numeric data processor for 86/30 board*
- o Intel iSBC 589 DMA controller
- o Plessey PSM 512A (1/2MB) memory board
- o Plessey PSM 1MB memory board
- o Ciprico Rimfire 45A disk and tape controller board
- o Network processor and interface boards
- o Central Data Corp. 15-slot MULTIBUS backplane and chassis model #CD31/6000-G110 #C2000

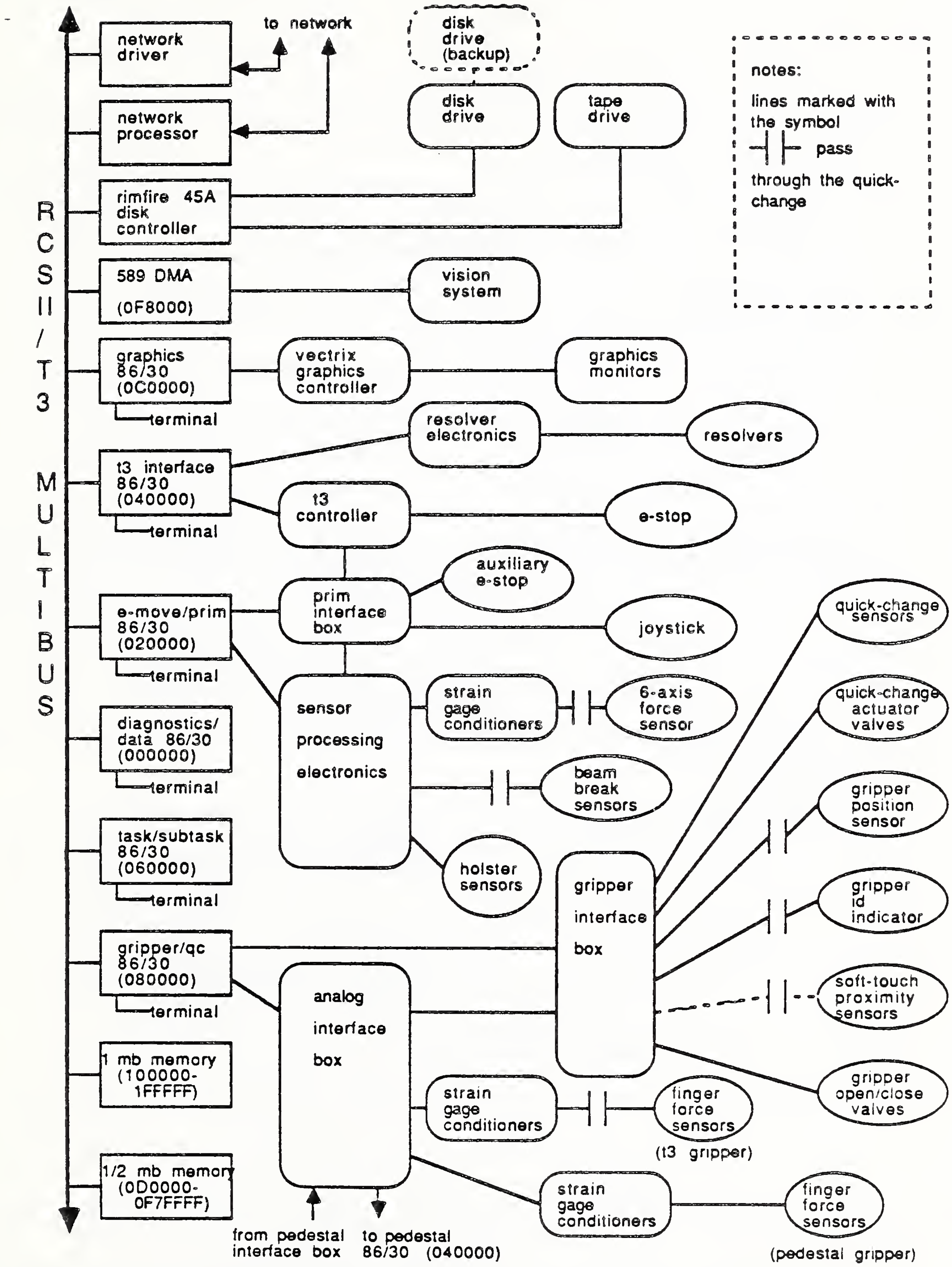


Figure III.1. RCS-II/T3 Electronic Hardware Components

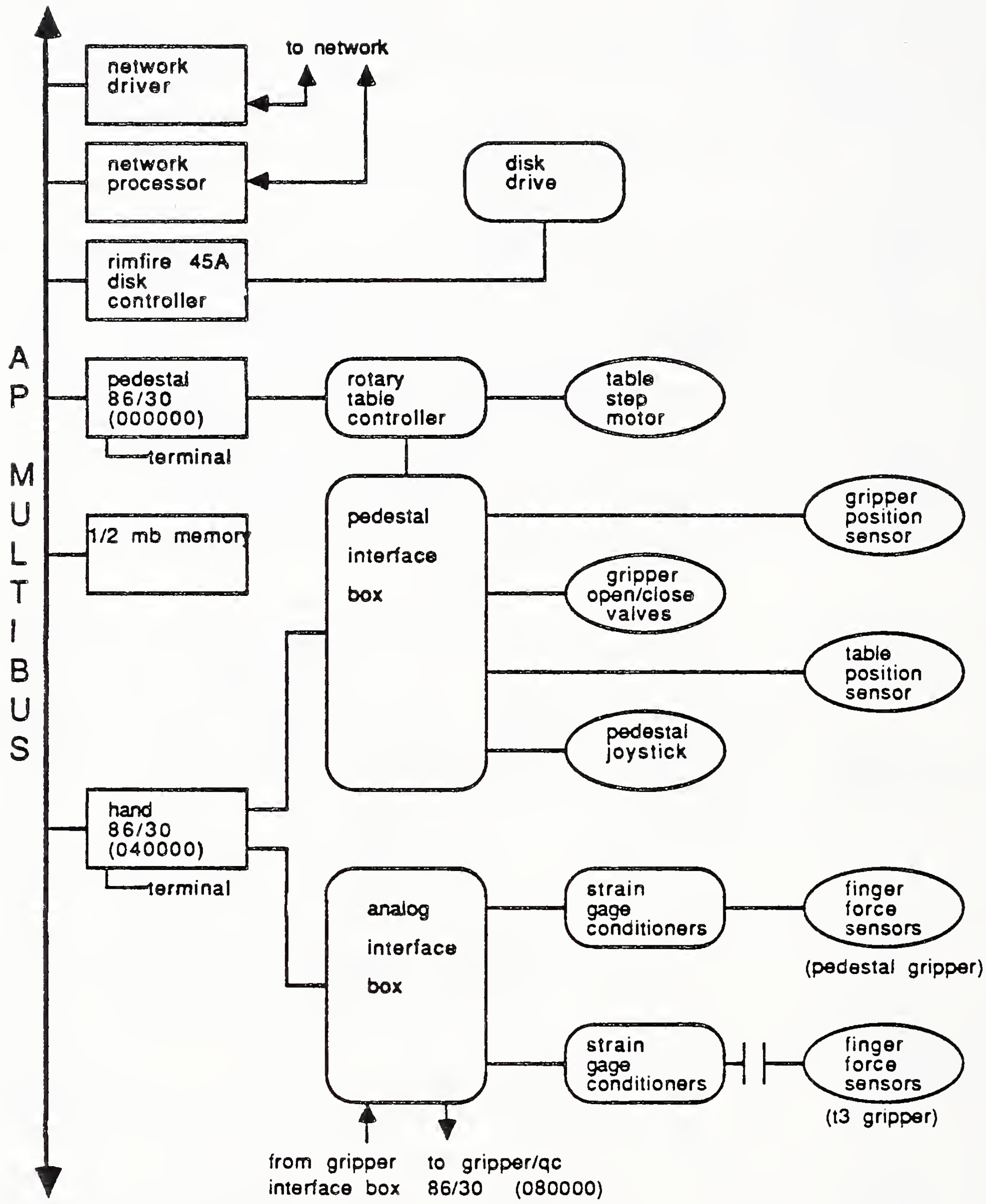


Figure III.2. Active Pedestal Electronic Hardware Components

- o Priam Diskos 6650 33-megabyte Winchester disk drive
- o Priam Diskos 3350 33-megabyte Winchester disk drive (backup)
- o Priam 3 00106-04 ANSI-standard disk interface adapter boards
- o Cipher F880 1/2 in. tape drive
- o Televideo 950 and Datamedia Elite 1521A terminals
- o Integral Data Systems Paper Tiger printer*
- o Vectrix graphics controller
- o Vectrix and Electrohome graphics displays

*not shown in Figure III.1

Figure III.1 also shows the Multibus address of the boards. All of the above items except for the terminals, printer, and graphics controller and display are located in the RCS-II/T3 robot controller rack.

2. <u>Active Pedestal Computer Hardware</u>

The Active Pedestal computer hardware (see Figure III.2) consists of:

- o Intel iSBC 86/30 single-board computers
- o Intel iSBX 311 analog input multimodules*
- o Intel iSBX 350 parallel I/O multimodules*
- o Intel iSBX 351 serial I/O multimodules*
- o Plessey PSM 512A memory board
- o Ciprico Rimfire 45A disk and tape controller board
- o Network processor and interface boards
- o Central Data Corp. 15-slot MULTIBUS backplane and chassis model #CD31/6000-G110 #C2000
- o Priam Diskos 3350 33-megabyte Winchester disk drives
- o Priam 3 00106-04 ANSI-standard disk interface adapter

boards

- o Televideo 950 terminal

*not shown in Figure III.2

The board addresses for the Active Pedestal controller are also shown in Figure III.2. The Active Pedestal computer hardware (except for the terminal) is located in the Active Pedestal controller rack.

3. RCS-II/T3 Sensor Hardware

Figure III.1 also illustrates how the various sensors associated with the T3 are connected to the computer hardware.

3.1. Vision System

The vision system consists of a camera mounted near the wrist of the T3, a line flash box, also mounted on the T3, flash electronics, located in the sensory processing electronics rack, and the vision computer system, which is housed in its own rack. Details of the system may be found in the AMRF vision system documentation. Vision requests and data are communicated over a high-speed parallel interface between the vision computer system and the 589 DMA board in the RCS bucket.

3.2. Resolvers and Resolver Electronics

The T3 is equipped with a redundant set of resolvers to measure the robot's position independently of the Acramatic feedback loops. The resolver outputs are connected to signal processing hardware located in the lower part of the Active Pedestal computer rack. The resolver electronics convert the resolver readings into 17-bit, integer-format joint positions. The joint positions are read by the T3 Interface board (board 4) through the on-board parallel I/O port.

3.3. Joystick Box

The Joystick Box, shown in Figure III.3, is used to provide manual control of the robot while it is under RCS control. The Joystick Box is connected to the Prim Interface Box, located at the bottom of the RCS-II/T3 computer rack. The Prim Interface Box sends the analog signals to the analog I/O multimodule on the Prim board (board 2) and sends the digital signals to the parallel port on the same board. A very long multiconductor cable, which attaches to a connector on the floor near the terminals, is used to connect the Joystick Box to the Prim

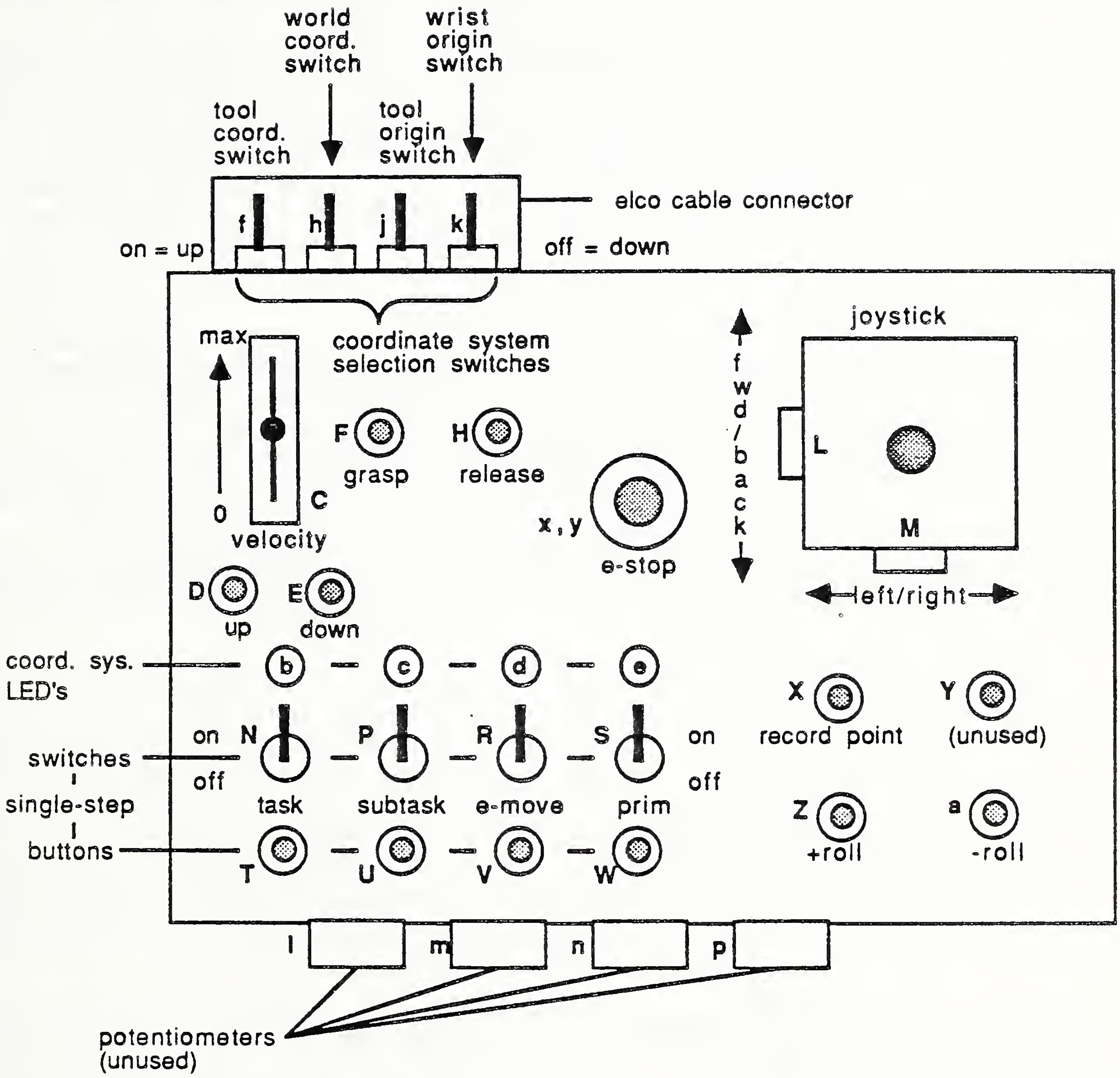


Figure III.3. RCS-II/T3 Joystick Box

Interface Box. Care must be taken to ensure that the carts of the Material Handling Workstation do not run over the Joystick Box cable. The Joystick Box is usually kept near the terminals.

Referring again to Figure III.3, the potentiometer C controls the velocity of the T3, whether the movements result from joystick manipulation or commands from the higher control processes. The emergency stop button x,y is connected to the T3 Emergency-Stop (E-Stop) circuit and causes the T3 hydraulics to shut down (with the resulting droop) when pressed. The joystick L,M controls the X and Y motions in the world and tool coordinate frames (see Figure III.4). The pushbuttons D and E control tool and world Z axis movements, and Z and a control + and - roll about the tool X axis. Pushbuttons F and H are used to open and close the gripper.

The four switches across the top of the joystick (near the ELCO connector) are used to select the coordinate system to be used for manually-controlled motions. Switch f selects world coordinates (translation), switch h selects tool coordinates (translation), switch j selects the tool origin (pitch and yaw rotations), and switch k selects the wrist origin (pitch and yaw rotations). Only one of the four coordinate switches may be active at any time. Switches are active when positioned up (away from the ELCO connector). The LED's b, c, d and e indicate which coordinate system is in use.

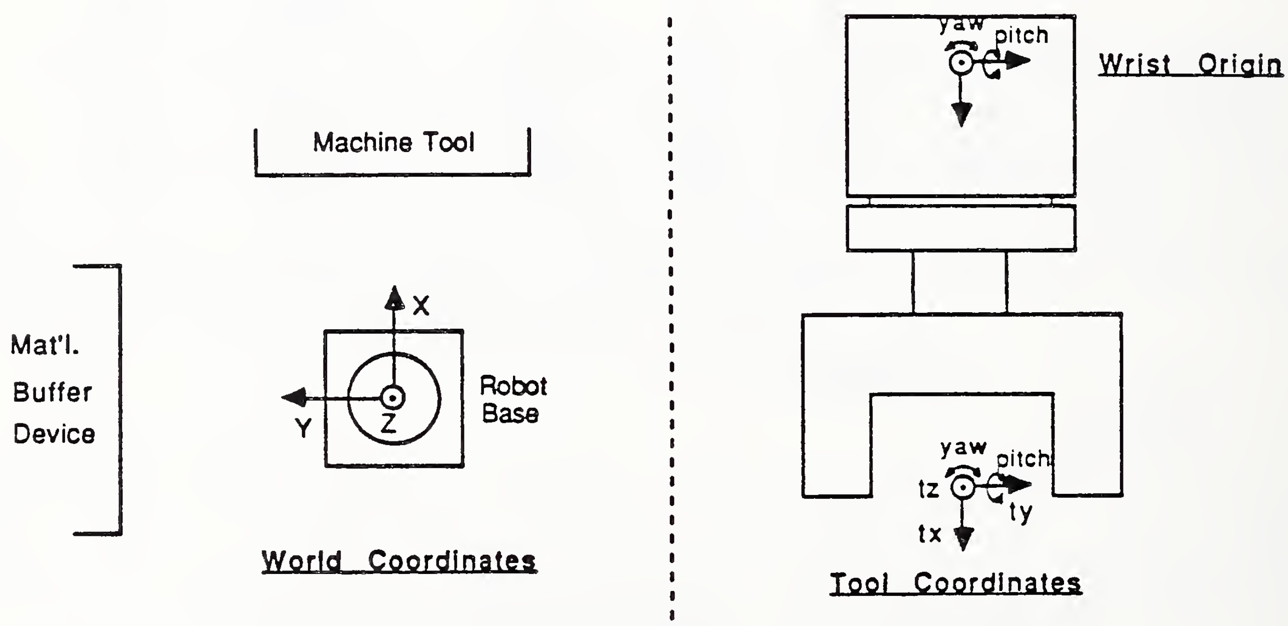


Figure III.4. Joystick Box Coordinate Frames

The pushbutton marked X is used to record poses. After the robot has been positioned as desired the button is pressed, which causes the pose coordinates to be displayed on the terminal connected to the Prim board. Recording a point also suspends execution of the Prim and E-move processes until a carriage return is entered. Pushbutton Y is an unused at this time.

The group of four switches and four pushbuttons in the lower left-hand corner of the joystick may be used to single-step the Task, Subtask, E-move and Prim processes. This is a very useful capability for debugging and for refining poses and trajectories. To single step a process, the corresponding switch is flipped toward the connector side of the box, which suspends execution of the process. With the switch set, pressing the corresponding pushbutton will cause the process to be executed once. Switches N and T control the Task process, P and U control Subtask, R and V control E-move and S and W control Prim. The Prim and E-move processes are the most often single-stepped. When Prim is single-stepped, individual knot points of a trajectory will be executed. When E-move is single-stepped, the robot will move between the intermediate points which have been defined for the trajectory.

The potentiometers l, m, n and p are not used at this time.

3.4. Holster Sensors

Each of the three gripper holsters [7] is equipped with four inductive proximity sensors to detect the presence and position of a gripper. The sensors are binary devices, positioned to be triggered by the "ears" of a gripper. There are two large-diameter sensors toward the back of each holster which are used to sense when a gripper mounted on the T3 is close enough to the holster to be released. The two small sensors at the front of each sensor have a much shorter detection range, and are used to determine when a gripper is properly seated. All four sensors are used to check whether a holster is empty or full.

The holster sensors are connected to the sensor processing electronics box, which is located in the sensor processing electronics rack. The sensor electronics box communicates the holster sensor signals to the Prim board through a parallel port multimodule.

3.5. Quick Change Sensors

The Quick Change [7] has two optical-reflectance type proximity sensors for determining whether the Quick Change is locked, unlocked, or stuck. The binary sensors detect the position of an

aluminum flange attached to the Quick Change locking ring. The Quick Change sensors are discussed more fully in reference [7].

The Quick Change sensor signals are routed to the gripper interface box, which in turn passes them to the parallel port on the gripper board (board 8). The gripper interface box is located in the sensor processing electronics rack.

3.6. Prismatic Part Gripper Sensors

The prismatic part gripper is equipped with a variety of sensors. There is a potentiometer that measures the position of the fingers and strain gages in each finger that measure the grasp forces. A gripper id "sensor" consisting of shorted Quick Change connections provides gripper identification (the prismatic part gripper is gripper 1). In addition, an array of eight beam-break sensors is available for detecting part edges. The gripper fingers each also house an analog-output optical reflectance "soft touch" proximity sensor, although these sensors are not connected currently, due to a lack of available Quick Change electrical connections. The signals of all the other sensors on the prismatic part gripper pass through the Quick Change, as indicated in Figure III.1.

The gripper position potentiometer, id indicator, and beam-break signals pass through the gripper interface box on the way to the gripper board. The gripper id then goes directly to the gripper board parallel I/O port. The analog gripper position signal is sent from the gripper interface box to the analog interface box, and then to the analog I/O multimodule on the gripper board.

The finger-mounted strain gages that measure grasp forces are connected, through the Quick Change, to strain gage amplifier/signal conditioners mounted in the sensor processing electronics rack. From there, the force signals are routed through the analog interface box to the analog I/O multimodule on the gripper board.

3.7. Cylindrical Part Gripper Sensors

The cylindrical part gripper only has finger position and gripper id sensors. These use the same Quick Change connections as the similar sensors on the prismatic part gripper, and therefore follow the same signal processing route to the gripper board. The cylindrical part gripper has the id number 2.

3.8. Tool Gripper Sensors

The gripper used for changing tools is equipped with a six-axis

force/moment sensor in addition to finger position and gripper id sensors. The finger position and gripper id signals again follow the same signal paths as with the other grippers. The tool gripper is gripper number 3. Only four axes of the six-axis force sensor are currently connected. These measure forces in the tool X, Y and Z directions (sensor Z, -Y and X directions, respectively). In addition, the strain gages that measure the moment about the tool Y axis are also connected. These signals are monitored during the tool change operation to provide disaster sensing. See reference [12] for further details.

The force signals are routed through their own dedicated quick change connectors to strain gage signal conditioners, after which they go to the sensor processing electronics box. The final destination for these signals is the analog I/O multimodule on the Prim board. Since this multimodule is connected to the Prim interface box, these force signals are routed through the Prim interface box, as well.

4. T3 Controller

RCS-II/T3 controls the T3 through the Cincinnati Milacron (CM) robot controller. The RCS-II/T3 control system sends commands to the CM controller in the form of Cartesian coordinates and Euler angles every 40 msec. The Cincinnati Milacron controller performs inverse kinematics and joint servoing for the robot. The CM controller is also used (with the CM teach pendant) to move the robot around when it is not under RCS-II/T3 control. Chapter IV will provide enough information about the CM controller to bring up the robot and connect it to RCS-II/T3. Users of the T3 robot are encouraged to read ref. [14] for further details on the CM controller.

RCS-II/T3 communicates with the CM controller via a Dynamic External Path Control (DEPC) interface. The T3-Interface process on board 4 handles the communications protocol for the DEPC link. DEPC communications take place over an RS-232 serial line connected between a port on the CM controller and a serial I/O multimodule on board 4.

5. PRIM Interface Box

The Prim Interface Box interfaces most of the digital and analog signals that are needed at the Prim level to the Prim/E-move board. This includes all joystick signals and the force signals from the sensor on the tool gripper. The Prim Interface Box is connected to the Prim/E-move board by two ribbon cables; one for digital signals which goes to the board parallel port, and one for the analog signals which goes to the analog I/O multimodule.

The Prim Interface Box also interfaces the Joystick Box E-Stop button to the T3 E-Stop stop circuit. Since the T3 E-Stop circuit operates on 115 VAC, the Prim Interface Box includes a solid-state relay to allow the joystick box E-Stop to operate at 5 VDC. An auxiliary E-Stop button, located near the terminals, is also connected to the 5 VDC Prim Interface Box E-Stop circuit. The Prim Interface Box is located below the RCS-II/T3 cardcage in the robot controller rack.

6. Emergency-Stops

The T3 E-Stop circuit is an extremely important part of the system in terms of safety. There is an E-Stop button at each of the following locations:

- o The front panel of the CM controller
- o On the CM teach pendant
- o Near the terminals used with RCS-II/T3
- o On the Joystick Box
- o Near the Active Pedestal

All of the E-Stops operate on a 115 VAC normally-closed circuit except for the E-Stop on the Joystick Box and the one near the terminals, which are 5 VDC, normally closed. Pressing an E-Stop immediately shuts off hydraulic pressure to the robot actuators. The E-Stop buttons must be used with extreme caution since the robot will drift down under its own weight without hydraulic pressure.

It is extremely important to remember that <u>HITTING AN E-STOP ALLOWS THE ROBOT TO DROOP</u>, which can cause extensive damage to the robot and other equipment. <u>IMMEDIATE ACTION MUST BE TAKEN AFTER HITTING AN E-STOP TO ENSURE THAT THE ROBOT DOES NOT DAMAGE ITSELF OR OTHER EQUIPMENT WHILE SETTLING</u>. This means that the robot shot pin [14] must be checked, the robot position observed, and a judgement must be made as to whether any damage will occur if the robot is allowed to settle. If damage is likely (or occuring!) the robot hydraulics must be powered back on <u>AS SOON AS POSSIBLE</u>. This is done by pressing the red Fault Reset button on the CM controller (it will be lit after an E-Stop), turning the Hydraulics keyswitch to enable, and pressing the Manual On/Off pushbutton (this is the same sequence used to bring up the robot under normal conditions--see the following chapter). Powering up hydraulics must also be done with caution, as the

distal end of the robot jumps up about 10 in. when hydraulic pressure is first applied.

7. Sensor Processing Electronics Box

The Sensor Processing Electronics Box (SPEB) interfaces signals to the parallel I/O multimodule on the Prim/E-Move board. The SPEB houses power supplies and signal conditioning electronics for the beam break and holster sensors. The SPEB also buffers the analog signals from the force sensor on the tool gripper and passes these signals to the Prim Interface Box. The SPEB is located in the sensor processing electronics rack.

8. Gripper Interface Box

The Gripper Interface Box interfaces all gripper and Quick Change sensor signals, except for force signals, to the Gripper board (board 8). It also provides power for the pneumatic valves used to control the gripper and the Quick Change actuators. The Gripper Interface Box is located in the sensor processing electronics rack. A single ribbon cable connects the Gripper Interface Box to the parallel port on board 8.

9. Active Pedestal (AP) Sensor Hardware

The Active Pedestal sensors are discussed in the following section.

9.1. Table Position Sensor

The rotary table of the Active Pedestal is equipped with a potentiometer mounted in the base to provide table position feedback. The potentiometer is also used to sense the limits on rotation of the table. The table position signal is sent to the Pedestal Interface Box, located in the sensor processing electronics rack. The Pedestal Interface Box passes the signal to the Analog Interface Box and also uses it to generate limit signals, which are sent to the Rotary Table Controller.

9.2. AP Gripper Sensors

The Active Pedestal gripper is nearly identical to the prismatic part gripper used with the T3. The Active Pedestal gripper, however, only has finger position and force sensing. The position potentiometer signal goes to the hand board via the Pedestal Interface Box and the Analog Interface Box, while the force signals are routed through strain gage conditioners and the Analog Interface Box. The AP gripper sensor signals are all passed from the Analog Interface box to an analog I/O multimodule

on the Hand board.

10. AP Rotary Table Controller

The AP rotary positioning table is driven by a step motor, whose controller is located at the top of the sensor processing electronics rack. The Rotary Table Controller receives commands from the Pedestal board (board 0) over an RS-232 serial link, driven by a serial I/O multimodule.

11. Analog Interface Box

The Analog Interface Box is used to buffer analog signals between the sensors and the computer boards. The inputs to the Analog Interface Box include the signals from the finger force sensors on both the T3 prismatic part and the Active Pedestal grippers. The Analog Interface Box also receives gripper position signals from both T3 and Active Pedestal grippers and the table position signal from the Active Pedestal. The signals associated with the T3 are sent to the analog I/O multimodule on the T3 Gripper board. The Active Pedestal signals are routed to the analog I/O multimodule of the Active Pedestal Hand board.

12. Strain Gage Conditioners

The strain gage amplifier/conditioners used are Vishay Instruments 2100 system units.* The offsets and gains have been properly set and should not require adjusting unless a strain gage is damaged or replaced. In this event, the reader is referred to reference [13] for a complete description of the conditioners and adjustment procedures.

IV. USING RCS-II/T3

This chapter contains the information required to run the control system and the robot. Instructions for starting and shutting down the system, editing and loading code, connecting to external systems, and running the robot are all included.

1. SYSTEM POWER UP

This section describes the power-up procedure for the equipment related to RCS-II/T3 and the robot. The procedure is as follows:

1. Turn on the power strip for the hallway vision system video displays and the control system graphics display. This power strip is located near the hallway vision displays. Two closed-circuit video cameras used for monitoring the T3 from the terminal area are also plugged into this power strip.

2. Press the CONTROL POWER ON pushbutton on the Cincinnati Milacron (CM) T3 controller console. The CRT will warm up (after a short while) and display the message ERROR-E22, and the red FAULT RESET button will light. The FAULT RESET button may not be pressed to reset until the Prim Interface Box is powered up (step 5).

3. Turn on the Daedal step motor controller at the top of the sensor processing electronics rack (if it is desired to run the AP). The mode switch should always be in the PROG position when the controller is turned on. Follow the startup instructions on the front panel of the controller, which says to:

   o Set the mode switch to PROG

   o Turn on the power

   o Set the mode switch to MANUAL

   o Push the RUN+ pushbutton until the table hits the + limit and stops

   o Push the RESET to set the counter to zero

   o Push the RUN-, then STEP+ and STEP-, as required until the counter display reads -20.00

   o Set the mode switch to PROG

- o Turn the power off, then back on again (this seems to be necessary to ensure that the controller gets initialized properly)

4. Turn on the power strip in the rear of the AP controller rack. This applies power to the disk and the cardcage or "bucket". Make sure the disk and fans are spinning. If the front panel LED on the bucket is not lit, turn on the bucket power by turning the lower keyswitch. Close the cabinet doors so the fans will be effective.

5. Turn on the power strip in the rear of the RCS-II/T3 controller rack. This supplies power to the disk, the bucket, and the Prim Interface Box. Again, make sure the disk and fans are spinning and turn on the bucket power if the LED is not lit. Be sure to close the cabinet doors.

6. Turn on the three power strips in the terminal area.

All the necessary equipment should now be powered up.

2. BASIC RCS-II OPERATIONS

This section will outline how to use RCS-II/T3 in sufficient detail to run the system either stand-alone or as an integrated part of the AMRF. Many of the operations are similar to those described in Chapter 6 of reference [1], which should be read prior to reading this section.

Note that errors made when typing in entries may be corrected using the RUB OUT (Datamedia terminals) or DELETE (Microterm terminals) key.

2.1. Starting the System

After the system has been powered up, RCS-II/T3 can be booted from the terminal connected to board 0 (labeled terminal 1). The steps to follow for the system boot are:

1. Enter SHIFT F1 (that is, press the SHIFT key and hold it down while pressing the F1 function key). The system response should end with "ok". If it does not, repeat until it does.

2. Enter SHIFT F2. This boots board 0. Wait for a screenful of text to appear, ending with a :R prompt.

3. Enter SHIFT F3. This "RUN-RCS" command boots the rest of the system remotely from board 0, which takes about 5

minutes. During this time you will see various screen displays on the terminals connected to the boards being booted. When the :R prompt returns to terminal 1 the entire system is booted. The code which is now loaded on the boards is that which was compiled at the time of the last system reload (see section 2.10), plus any changed code which is loaded from the "28 LOAD" block for the board (see section 2.8).

All boards should now have an :R prompt and all RCS-II/T3 operations except printing, taping, and reloading may now be performed.

2.2. Switching Between Boards on a Terminal

Some of the RCS-II/T3 terminals are used to communicate with more than one computer board. These terminals have switch boxes which are used to select the desired board. The boards connected to each terminal are listed below:

Terminal 1 (Microterm)	board 0	Diagnostics, data, system sychronization
Terminal 2 (Datamedia)	board 2	E-move, Prim processes
	board 8	QC, Gripper processes
Terminal 3 (Datamedia)	board 6	Task, Subtask processes
	board 4	T3 Interface process
	board 12	Graphics process

2.3. Editing a Block of Code

A knowledge of how to use the RCS-II/T3 screen editor is necessary to do anything useful with the system, including running the robot. The details of using the screen editor are covered in section 6.8 of reference [1]. The only deviation from the procedures presented there is that for the Datamedia terminals (terminals 2 and 3) the command to exit the screen editor is ^Q instead of ^R. Before editing code on a board, any routines executing on that board must first be stopped by entering ^C.

2.4. Loading Code

Loading code in RCS-II/T3 is similar to loading code in RCS or FORTH. To load a block of code, enter [block#] LOAD. It is necessary to load a block after it has been edited to redefine the routine(s) which have been altered. Routines can be edited and loaded without any other reloading or relinking, since the

SMACRO compiler is incremental. If a routine or variable declaration extends over more than one block, it is extremely important to load all the blocks in sequence. Otherwise, the routine will not be properly defined andunpredictable behavior may result. See section 6.5 of reference [1] for more information on the SMACRO compiler.

2.5. Changing Vocabularies

Each board contains a system vocabulary, called SDEF, as well as vocabularies for the processes resident on the board. To locate code on the system, you must be in the correct vocabulary on that board. To enter a vocabulary, simply type the name of the vocabulary. For example, entering SDEF on any board will set the current vocabulary to the system vocabulary. The vocabulary names for the primary processes are TDEF, STDEF, EDEF, and PDEF. Vocabulary names are given in the discussion of each of the processes in Chapters V and VI. The vocabulary into which a routine is compiled is named at the top of the block containing the routine.

2.6. Changing Modes

RCS-II/T3 users interact with the system through two modes: Run and Show. Run mode is used for executing routines and loading blocks of code. To execute a routine in Run mode, simply enter the name of the routine. The Run mode is the most commonly used mode, and is selected by entering the Run mode prompt, :R. In Run mode the word GO may be entered on boards 2, 4, 6, and 8 to execute the RCS-II/T3 processes. On board 0 the word SSGO is used in Run mode to execute the Diagnostics process and generate the system synchronization pulse.

Show mode is used primarily to look at and change the values of variables interactively. To enter the Show mode, simply type the SHow mode prompt, :S. In Show mode, the type and value of a variable is displayed when the variable name is entered. The value of a variable thus displayed may be changed by entering <= [new value]. Of course, the new value must be the proper type for the variable.

One use of the Show mode is in the definition of special show words which display the values of groups of important variables when they are executed. There are show words which display the values of the input and output interface variables for each of the Task, Subtask, E-move, and Prim processes. You must be on the proper board to use these words, which are:

TAI - to display the Task input command interface from the

HWSC and output status to the HWSC (board 6).

TAO - to display the Task output command interface to Subtask and input status from Subtask (board 6).

SB - to display the Subtask interfaces to Task and E-move (board 6).

EM - to display the E-move interfaces to Subtask and Prim (board 2).

II - to display the Prim interfaces to E-move and the T3 Interface process (board 2).

Sometimes the prompt :C may appear when a block is loaded. This means that the system is in Compile mode (which is usually transparent to the user). The reason that the system is still in Compile mode instead of returning to Run mode is usually that the routine which was loaded spills over into subsequent block(s) which must also be loaded.

2.7. Locating Source Code

The RCS-II/T3 disk is organized into sections containing 1000 blocks each. The source code for each RCS-II/T3 process is contained in one of these sections. The location of the 1000 block areas is specified by the FORTH variable OFFSET. To set the current offset, enter <u>[offset#</u> <u>OFFSET</u> <u>!</u> just as you would in FORTH. For example, for the Task process, located at blocks 16000-17000, enter <u>16000</u> <u>OFFSET</u> <u>!</u>. Once the offset is set, block contents may be looked at by using the FORTH word LIST. The block is displayed by entering <u>block#</u> <u>LIST</u>. For most processes, a directory/load block system similar to that described in chapter 6 of reference [1] is used. The location of each of the RCS-II/T3 processes is:

Task	16000-16999
Subtask	17000-17999
E-move	18000-18999
Prim	12000-12999
T3 Interface	19000-19999
Parts gripper	23000-23999
Quick change	25100-25199
V-gripper	25200-25299
Tool gripper	25300-25399
Diagnostics	24000-24999
Graphics	27700-27999

The LOC command (reference [1], section 6.4) may be used to find

the block where a specific word is defined.

2.8. Using 28 LOAD Blocks

On each board there is a block which is used for initializing the processes on the board and the board I/O hardware. This initialization block is also used to automatically load routines which have been changed since the system was last reloaded. This block is block 28 at the offset for one of the processes on the board. The offsets of the "28 LOAD" blocks are:

board 2	12000 OFFSET
board 4	19000 OFFSET
board 6	16000 OFFSET
board 8	25000 OFFSET

The 28 LOAD block for each board is automatically loaded during the RUN-RCS part of the system boot. To have a block of code loaded from a 28 LOAD block, you must edit block 28 and add [offset] [block#] OLOAD, where the offset is that which contains the block# which is to be loaded.

2.9. Using Printing and Tape Backup Utilities

Terminal 2 is used to make backup tapes and printouts of the RCS-II/T3 source code. A different system boot procedure must be followed to use these utilities. The procedure:

1. Power up the RCS-II/T3 bucket and the terminals. If the RCS-II/T3 system has already been booted using the normal procedure, the bucket should just be reset by turning the upper keyswitch. The printer (connected to terminal 2) should also be powered on. If a tape is to be made, the tape drive (located in the RCS-II/T3 controller rack) must be powered on.

Note: The system response to each of the following commands should be "ok".

2. Select board 2 on terminal 2 and enter HEX F7 D7 OUTPUT.

3. Enter 26F0 30 ERASE.

4. Enter 2 CBOOT 2 MBOOT. A screenful of text should appear.

5. Enter 3 CUSTOM. This loads the printing and tape utilities.

The system is now ready to execute the utilities. The printing utility CPRINT prints ten blocks onto a single page. To print

out blocks, first hit the PRINT ON key on the terminal, which should cause the PRINT LED to light. Since this key also sends an undesired character, hit the ERASE key or RETURN before continuing. Next, make sure the printer is properly set up and enter [block# block#] CPRINT, where the first block# is the first block to be printed and the second block is the final one to be printed.

To make a system backup on tape, first load a tape into the tape drive, making sure the write enable ring has been removed. Press the LOAD/REWIND and ON-LINE buttons on the tape drive. After the tape drive LED's stop flashing, the tape may be written to by loading block 297 at offset 1000. This block uses the ADD-TO-TAPE command to write RCS-II/T3 files to the tape, as listed in the block. When the system returns "ok" you may rewind the tape by pressing the UNLOAD button on the tape drive.

To read a system backup tape in, block 292 at offset 1000 may be used. This should only be done when absolutely necessary, as reading in a tape overwrites the current information on the disk. After a system backup tape is read in, the system will have to be reloaded as described in the next section.

2.10. Reloading the System

The entire system may be reloaded to update the compiled version of the RCS-II/T3 code. The entire procedure is performed from terminal 0 (through board 0). To reload the system, you must:

1. Reset the RCS-II/T3 bucket.

2. Boot board 0 by entering HEX F7 D7 OUTPUT, 26F0 30 ERASE, 0 CBOOT 0 MBOOT.

3. Enter 1 CUSTOM.

4. Enter NQUIT

5. Enter NEW-MOLE2

6. Enter 24000 OFFSET ! 2 LOAD.

7. Enter SAVE

8. Enter 1360 DO-BUCKET. Wait a long time, while code on all boards is loaded and saved.

9. Enter 21000 OFFSET ! 905 LOAD 906 LOAD.

10. Enter 20000 OFFSET ! 0 LOAD.

11. Enter SAVE.

This completes the system reload procedure.

2.11. Shutting Down the System.

If the robot is running, it must of course be shut down before the rest of the system (see section 3.4).

To shut down the rest of the system, any executing processes should first be stopped by entering ^C on each board. Then the command WUNLOAD should be entered from one of the terminals. This will shut down the disk, and should always be entered before the RCS-II/T3 controller rack power is turned off. The power strips for the terminals and the RCS-II/T3 controller rack can then be turned off. Turn off the power strip for the hallway video monitors and cameras.

The shut down procedure for the Active Pedestal may be found in section 5.3.

3. RUNNING THE ROBOT

This section describes the steps to follow in order to run the T3 under RCS-II/T3 control.

3.1. Robot Safety Precautions

The T3 is a very large, powerful, and potentially dangerous machine. ANYONE WHO USES THE ROBOT SHOULD HAVE EXTENSIVE SAFETY TRAINING AND BE EXTREMELY FAMILIAR WITH THE CINCINNATI MILACRON OPERATING MANUALS (see reference [14]) AND THE SAFETY WARNINGS THEY CONTAIN.

It is necessary to use extreme caution at all times when the robot hydraulics are activated. In this state the robot can move unexpectedly in any direction with surprising speed if there is a hardware or electrical malfunction. NO PERSON SHOULD EVER BE IN THE ROBOT WORK VOLUME WHEN THE ROBOT HYDRAULICS ARE POWERED ON. The robot work volume is clearly marked on the floor with hazard tape, and includes an allowance for the longest gripper.

The E-stop buttons should always be very close at hand. Depressing an E-stop button will automatically turn off hydraulic power to the arm, but will allow the control console to remain operational. An error code will appear on the CRT screen and the FAULT RESET indicator will be illuminated.

CAUTION: If the robot arm is not in the HOME or SAFE position when the E-stop button is depressed, the arm could drop. Failure to observe this caution may result in damage to or destruction of valuable equipment.

To restart the system, depress the FAULT RESET pushbutton to cancel the error. Turn the HYDRAULICS keyswitch to the ENABLE position so the HYD ENABLED light is illuminated. Depress the MANUAL ON/OFF pushbutton to enable hydraulics and put the control in the manual mode of the CM controller.

ALTERNATIVE MEANS OF STOPPING THE ROBOT

Because of the danger associated with activating an E-stop, it is preferable to stop the robot by ramping the Joystick Box velocity down to zero or switch on the Prim single-step switch, if there is time. Either of these actions should stop the robot. If they don't, of course, then hitting an E-stop is the only alternative.

3.2. System Configuration Setup

This section describes how to set up RCS-II/T3 to run the robot in a stand-alone mode, which is always done before connecting to the HWSC. Assuming that the system has been booted as described in section 2.1, the next steps are:

1. On board 6, list block 903 at offset 16000 (it should be at this offset already.). This block contains the Task level preprocessing routines, and should look like that shown Figure IV.1 for stand-alone operation. In SMACRO (the language used for RCS-II; see reference [1]), the % symbol may also be used to temporarily prevent routines or parts of routines from being executed. This is the case in Figure IV.1, where, for instance, the routine READ-HWS-COMMAND is commented out so that the command from the HWSC will be ignored. Edit and load block 903 as necessary.

   When it is desired to execute a process without running the lower level processes, the communications routines to the lower processes are commented out and the lower-level-done routine for the process (usually called lld) is put in by erasing the % sign. This is often useful for testing and debugging purposes.

2. List block 907 at the same offset. Again, the block should have the routines that provide for communication to external

BLOCK# 903

```
TDEF   EXEC-0  PREPROCESS
\   r    READ-HWS-COMMAND
    r    READ-SUBTASK-STATUS
\   r    READ-DAS-STATUS
    r    Parse-das-status
\   r    lld
    r    DB-DONE                   \ PUT IN FOR NO DATA BASE
    r    TRANSLATE-HWS-CMD
    r    DETERMINE-REF-ACTION
    r    DETERMINE-TASK-STATE
    r    CHECK-IF-NEW-COMMAND
\   r    DIAGNOSTIC
    r    HWS-RCS-STATUS-PRINT
```

BLOCK# 907

```
TDEF   EXEC-0  POST-PROCESS

    r    SET-REF-ACTION
    r    ECHO-REF-COMMAND-VAL
    r    INC-COMMAND-#
    r    WRITE-SUBTASK-COMMAND
\   r    WRITE-HWS-STATUS
\   r    WRITE-DAS-COMMAND
\   r    UPDATE-MAILGRAM?
\   r    UPDATE-MAILGRAM->DAS-?
```

Figure IV.1. PREPROCESS and POST-PROCESS Routines for Stand-alone Operation (when RCS-II/T3 is not connected to HWSC)

systems commented out, as in Figure IV.1. Edit and load as required.

3. Enter 863 12 12 LINE-LOAD. This places a PAUSE command in the Task input buffer.

4. Enter GO. This enables the Task and Subtask processes to start executing. They will not begin cyclic execution, however, until the system synchronization pulse is started by entering SSGO at terminal 0 (board 0), which may be done at this time.

5. Blocks 903 and 907 should be checked for the Subtask, E-move, and Prim processes to make sure the desired system components will communicate with one another.

6. Enter GO on boards 4, 2 and 8. This starts the execution of the T3 Interface, the E-move/Prim and Quick Change and Gripper processes.

3.3. Powering up the Robot and Initializing the DEPC Link

The procedure for connecting the T3 to RCS-II/T3 is described in this section. This assumes that RCS-II/T3 has been brought up as described in the previous section. The required steps are:

1. Enter ^C on board 2 to stop the E-move and Prim processes. All other processes should still be running. Look at the control system graphics display and confirm that the Task, Subtask, E-move, and Prim processes all have PAUSE commands.

2. Enter .R on board 2. This executes the Prim RESTART initialization command. Check that the Joystick Box velocity is set to zero, the coordinate selection switches are all off, and that the Prim single-step switch is active. This is the safest state of the Joystick Box.

3. Switch to board 4 on terminal 3. There should be a display of the robot position which gets updated every cycle.

4. Power on the the CM T3 controller if this has not already been done.

5. Push the red FAULT RESET button if it is lit.

6. Check the robot work volume to make sure it is clear of people and foreign objects. Also check that the grippers are properly seated in the correct holsters.

7. Turn the HYDRAULICS keyswitch on the CM control console to ENABLE (the keys to the CM controller are kept in the drawer beneath terminals 2 and 3). The red warning lights on the robot and the HYD ENABLED light on the CM console should come on.

WARNING! THE NEXT STEP ACTIVATES THE ROBOT HYDRAULICS, PUTTING IT IN A "LIVE" AND DANGEROUS STATE. THE ROBOT WILL MOVE SEVERAL INCHES WHEN HYDRAULICS ARE POWERED UP.

8. With your right hand covering the control panel E-stop, press the MANUAL ON/OFF pushbutton, which turns on the robot hydraulics. After a short delay, the distal end of the robot will jump several inches and then stabilize.

9. The next step is to align the robot arm to fixed reference position call HOME. The message * = ALIGNED with the numbers 1 through 6 will be displayed on the CRT screen. Using the T3 teach pendant, located on a bracket on the Vertical Workstation side of the Kardex buffer storage system, move the base, shoulder, and elbow joints to align the white arrows on each joint. The yaw joint should be aligned such that the gripper projects staight down or out (in terms of left to right). The pitch joint should be positioned such that an imaginary line extended through the gripper axis intersects the elbow joint. The roll joint should be rotated so the camera is at the top of the wrist. As each axis is aligned, an asterisk will appear under the number corresponding to each axis (axes are numbered serially from the base).

Check that all axes have asterisks. Note that some joints do not have a unique position where the asterisk comes on. It is, therefore, important to position the axes as described above to make sure the robot is at the correct position.

CAUTION: The next step will again cause the robot to jump.

10. Press the AUTO push button on the CM control console. The robot will move directly to the HOME position.

11. Check the robot position display on terminal 3. The displayed values of the frame-feedback variable should be approximately:

    0 -7145 -1483 0 -2123 -2119 3000 0 0

    If any of the values is off by more than 4, or has an incorrect sign, then the resolver electronics (located at the bottom of the Active Pedestal controller cabinet) must be reset according to the following instructions:

    a. Make sure the Resolver Electronics Box is powered on (turn on the power strip in the rear of the cabinet if not).

    b. Switch the double MANUAL/AUTO switch to MANUAL.

    c. Set the three joint address switches as follows:

       4 - down 2 - down 1 - down

    d. Push the button marked joint no. 0.

    e. Set the three joint address switches as follows:

       4 - down 2 - down 1 - up

    f. Push the button marked joint no. 1.

    g. Set the three joint address switches as follows:

       4 - down 2 - up 1 - down

    h. Push the button marked joint no. 2.

    i. Set the three joint address switches as follows:

       4 - down 2 - up 1 - up

j. Push the button marked joint no. 3.

k. Set the three joint address switches as follows:

4 - up 2 - down 1 - down

l. Push the button marked joint no. 4.

m. Set the three joint address switches as follows:

4 - up 2 - down 1 - up

n. Push the button marked joint no. 5.

o. Set the three joint address switches as follows:

4 - down 2 - down 1 - down

p. Switch the MANUAL/AUTO switch to AUTO.

q. Recheck the values on the screen; they should be ok now.

12. Press the CYCLE START button on the CM console. The robot will execute two short vertical motions and stop. The DEPC link should now be initialized.

Sometimes a "WD" error will appear on the console CRT screen after CYCLE START is pressed. If this happens, press the CM console INTERRUPT button, then the CANCEL key on the CM keyboard twice. Enter a new <u>.R</u> on board 2, press AUTO and then CYCLE START on the CM console again.

13. Enter <u>PRIM-LEVEL</u> <u>PRIM-LEVEL</u> <u>II</u> on terminal 2, still connected to board 2. This executes the Prim process twice, which should result in receiving the proper position feedback from the T3I process. To verify this, check the following variables (the proper value is given below the name, as on the terminal screen):

tp-xyz-input	tp-xyz-output
0 -7145 -1483	0 -7145 -1483
wr->tp-in	wr->tp-out
0 -2123 -2119	0 -2123 -2119
tp->fp-in	tp->fp-out
3000 0 0	3000 0 0

If the values are more than 4 off or have a different sign from those shown, follow the instructions for handling a WD

error, above.

CAUTION: The next step puts the robot under RCS-II/T3 control.

14. If the position variables are correct, enter GO on board 2, watching the robot with an E-stop close at hand. The robot SHOULD NOT MOVE when this is done. If it does, hit the E-stop IMMEDIATELY.

15. The robot is now under local RCS-II/T3 control, and should respond to the Joystick Box and commands entered at the Task level. To test the Joystick Box response, flip up one of the coordinate system selection switches (usually world coordinates), switch off the Prim single-step switch, ramp up the velocity about halfway, and use the Joystick Box controls to move the robot a short distance.

16. Set the velocity back down to zero and turn off the coordinate selection switch. Enter ^C on board 6 to stop the Task and Subtask processes.

17. Enter 863 3 3 LINE-LOAD on board 6. This enters a command at the Task level to move to the position defined as HOME position and is a safe place to power down the robot with any gripper on.

18. Enter GO on board 6. The command will be displayed on the terminal screen.

19. Ramp up the velocity on the Joystick Box. The robot should move a short distance back to the RCS-II/T3 HOME position.

    Note on the graphics display that the command and status boxes for each process change apropriately as a command is executed. When the command is finished, all the status boxes will read "done" and a "Finished" message will appear on the screen of terminal 3. Set the velocity back down to zero when the command is finished.

The robot may now execute commands entered into the Task command buffer. Blocks 863, 667, and 880-899 at offset 16000 contain commands for different parts and motions. The SMACRO word == stuffs the Task command buffer when it is loaded followed by the appropriate parameters. To execute commands from these blocks, the standard procedure is:

1. Enter ^C on board 6.

2. Enter [block#] [line#] [line#] LINE-LOAD, where the block#

and line# values apply to the desired command.

3. Enter GO on board 6.

4. Ramp up the velocity for the robot to execute the motion and set it back to zero when complete.

If the robot is to sit idle for a while, it is wise to turn on the Prim single-step switch to prevent accidental motion caused by disturbing the Joystick Box. Note that if a coordinate system switch is active and the Joystick Box velocity is non-zero, any movement of the Joystick Box motion controls will override the current command, and cause the robot to move as commanded by the Joystick Box controls.

To run RCS-II/T3 with commands from the HWSC, see section 4.1.

3.4. Shutting Down the Robot

To shut down the robot, first move it to the HOME position by executing an appropriate command as described above (for example, block 863, line 3 or block 667, line 10, depending on where the robot is). An E-Stop button may then be pressed to power down the robot hydraulics. WHENEVER THE ROBOT IS POWERED DOWN, YOU SHOULD CHECK THAT THE SHOT PIN UNDER THE SHOULDER HAS EXTENDED TO PREVENT THE ROBOT FROM DRIFTING TO THE FLOOR. The CM controller can then be turned off by pressing the CONTROL POWER OFF button.

In some situations, expecially if a problem has been encountered while the robot has been under RCS-II/T3 control, it is desirable (or unavoidable) to disconnect the robot from RCS-II/T3 before moving to HOME and shutting down. To disconnect the robot from RCS-II/T3 and return the robot to manual control under the CM teach pendant, press the INTERRUPT button on the CM console and then press the CANCEL key on the CM keyboard twice. The robot may now be moved around with the CM teach pendant to close to the HOME position before shutting down.

4. CONNECTING TO EXTERNAL SYSTEMS

The information needed to connect RCS-II/T3 with external systems is presented in this section.

4.1. Connecting to the HWSC

The procedure for connecting RCS-II/T3 to the HWSC once the robot is under RCS-II/T3 control is very simple. The robot should be at HOME and a PAUSE command (block 863, line 12) should have been

the last command before the HWSC is connected. The Data System should also be connected, if desired, before connecting to the HWSC (see section 4.2). Then, the steps are:

1. On board 6, edit block 903 at offset 16000 and take out the comment symbol so READ-HWS COMMAND will execute.

2. Load block 903.

3. Edit block 907 so WRITE-HWS-COMMAND and UPDATE-MAILGRAM? will execute.

4. Load block 907.

5. Enter <u>188</u> <u>LOAD</u>.

6. Enter <u>641</u> <u>LOAD</u>.

7. Enter <u>GO</u>

To disconnect from the HWSC, simply edit and load blocks 903 and 907 so the HWSC communications routines listed in steps 1 and 3 do not execute.

4.2. <u>Connecting to the Data System</u>

Currently, Data System connection, if desired, should take place before connection to the HWSC. The steps are:

1. Edit and load block 903 at offset 16000 on board 6 so READ-DAS-STATUS will execute and DB-DONE will not execute.

2. Edit and load block 907 so WRITE-DAS-COMMAND and UPDATE-MAILGRAM->DAS-? will execute.

3. Execute the STARTUP command (block 667, line 0) just like any other Task command. The terminal screen will display Aborting. . . , Initiating. . . , Connecting . . . and eventually, Finished, in sequence. When the command has finished, Data System connections have been successfully made.

To disconnect from the Data System, execute the SHUTDOWN command (block 667, line 1), and edit and load blocks 903 and 907 at offset 16000 so the Data System communication routines will not execute, but the routine DB-DONE will (as in Figure IV.1).

The Data System STARTUP and SHUTDOWN routines can also be issued from the HWSC, although this has not been done at present.

4.3. Connecting to the Active Pedestal

To communicate with the AP, edit and load blocks 903 and 907 at offset 17000 on board 6 so that the AP communications routines (READ-AP-STATUS, WRITE-AP-COMMAND, and UPDATE-MAILGRAM?) are executed.

4.4. Connecting to the Vision System

Vision system connections are automatically performed during the RUN-RCS boot procedure. To check that vision communications are in the E-move preprocess and postprocess, check blocks 903 and 907 at offset 18000 on board 2. READ-VISION-DATA and WRITE-VISION REQUEST should not be commented out if vision is to be used.

5. ACTIVE PEDESTAL (AP) OPERATIONS

This section describes the steps required to operate the AP with commands from RCS-II/T3.

5.1. Starting the System

First, the AP equipment must be powered up as described in section 1. Microterm terminal 4 is used to communicate with the AP controller. The AP controller operates on a version of RCS very similar to that described in reference [1], which will be referred to as RCS/AP. RCS/AP uses background tasks and master/slave operation for communicating with more than one board from a single terminal. The procedure for booting RCS/AP is as follows:

1. Enter SHIFT F1 on terminal 4. Should respond with "ok"; if not, try it again.

2. Enter SHIFT F2. Should respond with "ok".

3. Enter init-cm. Should respond with "ok".

4. Enter SHIFT F3. Should respond with :R ped>.

5. Enter BOOT-SYSTEM. Should respond with "ok".

6. Enter SHIFT F3. Should respond with :R hand>.

7. Enter 38 LOAD. Wait for the hand calibration routine to finish. Should respond with :R hand>.

8. Enter BGO. Should respond with :R hand>.

9. Enter ^X. Remember to press RETURN for this. Should respond with :R ped>.

10. Enter 38 LOAD. Wait for the rotary table calibration routine to finish. Should respond with :R ped>.

11. Enter CLP. This starts the Ative Pedestal controller execution loop; control will not return to the keyboard unless execution is stopped by entering ^C.

RCS/AP is now ready to accept commands from RCS-II/T3.

5.2. Shutting Down the Active Pedestal

To shut down the AP, follow these steps:

1. Enter ^C on terminal 4.

2. Enter WUNLOAD to shut down the RCS/AP disk.

3. Power down the terminal, RCS/AP controller rack, and rotary table controller in the sensor processing electronics rack.

V. PRIMARY CONTROL SYSTEM PROCESSES

This chapter describes the functions, commands, and interfaces of each of the RCS-II/T3 primary control system processes; the processes which control the motion of the T3. The code for each of these processes is organized into a 1000 block area on the disk. A directory/load block system similar to that described in section 6.3 of reference [1] is used. A template of a typical control process is located at offset 14000. This shows the arrangement of process components within the 1000 block areas. Usually, all variables for a process are defined in blocks 100-199. Preprocess routines are located in blocks 200-299. Routines called during command execution may be found in blocks 300-399 and 400-499. Blocks 500-599 are reserved for sensory request routines (such as E-move vision procedures). Postprocessing routines are in blocks 600-699. Blocks 700-799 and 800-899 are normally used for the decision processing routines of individual commands. The primary state table and routines used to execute the overall process are located in blocks 900-999. Each process has its own separate vocabulary in which all variables and routines are defined.

In the discussion of process interfaces, only the interface to the process above is described. Also, names of SMACRO data files will sometimes be used in the description of an interface variable; these files are discussed in Chapter VII. In the descriptions of how commands get decomposed, it should be remembered that RCS-II/T3 activities are based on the state of the system, rather than a static predetermined sequence. When commands are described as sequences of events or as producing a sequence of commands to subordinate processes, the sequence should only be regarded as the nominal procedure for accomplishing the command. The actual execution will depend on whether any subgoals of the command have been achieved, what sensory values are being observed, etc.

1. PRIM

The function of the Prim(itive) process is to decompose primitive motion commands consisting of Cartesian space straight-line path segments into trajectory points which are sent to the T3 controller (through the T3 Interface process) every 40 ms. The different input commands allow different termination conditions to be specified for path segments. For instance, some commands (GO-UNTIL, GO-WHILE) allow for termination based on a specified (desired or undesired) sensor condition. The robot may stop at the end of the segment (GOTO-PT), or it may keep moving if the segment is part of a multi-segment trajectory (GO-THRU). The motion may be a straight line path segment to a relative position

defined with respect to another position (DELTA-MOVE).

The actual trajectory interpolation functions are performed by an object code module, Traj-program, which is not described here. The Prim process routines are located at offset 12000, and the name of the Prim vocabulary is PDEF.

1.1. Commands

Prim input commands are the lowest-level commands that get decomposed by RCS-II/T3. In addition to the straight-line motion primitives mentioned above, the Prim process executes commands which perform sensor calibration and robot initialization functions.

Since the Prim process determines the position of the next robot goal point every 40 ms, this is where all low-level sensors which affect the robot motion on a cycle-to-cycle basis are brought in.

Each command has its own decision processing, and the execution of a command is controlled by the states of variables pertinent to the command. For most RCS-II/T3 processes, one of these variables is a "reference action" variable, which indicates the state of attainment of subgoals of the command. In addition, at the Prim level such variables as the distance to the goal point and the states indicated by sensors are used to control command execution.

GO-THRU command

The GO-THRU command is used to move the robot toward an intermediate trajectory point, where the robot will not stop at the point, but merely pass near it ("thru") on the way to the next point.

GOTO-PT command

GOTO-PT is the command used when it is desired to move to the goal position and stop, as at the end of a trajectory.

GO-UNTIL command

GO-UNTIL is the RCS-II/T3 implementation of a guarded move, in which the robot moves toward some goal point until a desired sensor condition is achieved. The command is successfully completed when the condition is reached, and a done status is sent to E-move. If the sensor condition is not achieved by the time the robot reaches the goal point, then an error has occurred, and the Prim process will not report done until the

situation is rectified.

DELTA-MOVE command

DELTA-MOVE is used to perform relative motions with respect to some fixed location. The relative motion is specified by a "movetable" (see Chapter VII) which is applied to the desired position.

CAL-SENSOR command

CAL-SENSOR is a command used to calibrate the six-axis force sensor prior to executing a GO-WHILE command. This is necessary to get a useful baseline reading of the strain gages which takes into account the weight of an object held by the gripper, and bias forces which result from a particular robot configuration. This command could perform the same function for GO-UNTIL commands as well, if desired.

GO-WHILE command

GO-WHILE is used when it is desired to move to a goal point, but abort if the designated sensor condition is detected. The effect of this command is to go-while-not-detected.

PAUSE command

The PAUSE command is used to command a process to do nothing. There is a PAUSE command defined for each RCS-II/T3 process. The PAUSE command at the Prim level is somewhat different from that at other levels, in that one function which can be performed with a PAUSE command from the level above is manual control of the robot with the Joystick Box.

ROBOT-INIT command

The ROBOT-INIT command issues a RESTART command to the T3 Interface level to initialize the DEPC link.

1.2. Interfaces

The input command interface to the Prim process consists of the following variables:

command-#-in
    integer; incremented each time a new command is sent.

"command"
    10-character string; contains the name of the input command.

"sensor"
10-character string; contains the name of the sensor to be monitored during a GO-WHILE or GO-UNTIL command.

^-pose
integer; points to the record in the POSE file which contains the current goal pose to Prim.

^-movtab
integer; points to the record in the MOVTAB file which contains the transformation to be applied to the current goal pose.

^-traj-para
integer; points to the record of the TRAJ-PARA which indicates the values of maximum acceleration, maximum velocity, etc. to use.

endeff-id#-in
integer; indicates the id number of the current gripper on the robot.

The output status from the Prim process to E-move consists of the following variables:

command-#-echo
integer; echos the number of the current command from E-move.

status-report
12-character string; contains the status reported to E-move. The possible contents are:

Status	Description
executing	The current command is executing.
done	The current command is finished executing.
not-detected	The desired sensor condition of a GO-UNTIL was not achieved.
joystick-on	The Joystick Box is active and controlling the motions of the robot.
t3-link-off, t3-link-err, no-resp-t3i	The T3I process reports a DEPC link error.

t3-X-command, The T3I process reports a bad command.

sensor-error A sensor overload or malfunction has been identified.

detected-err The sensor condition specified in a GO-WHILE command was detected on the way to the goal.

restart-status
integer; is a flag that indicates to E-move that a RESTART command is being sent to the T3I process.

pres-pose-^-out
integer; points to the record in the POSE file which contains the pose representation of the current position commanded from Prim to the T3 Interface.

sensor-pose-^-out
integer; points to the record in the POSE file that contains the pose representation of the current position command from the T3 Interface to DEPC.

detected-pose-^-out
integer; points to the record in the POSE file that contains the pose of the robot when the sensor condition specified in a GO-UNTIL command was detected.

hol-R-status
hol-L-status
hol-C-status
integers; contain the sensor values for the right, left, and center holsters.

joy-grip-status
integer; contains gripper open/close information from the Joystick Box.

2. E-MOVE

The E(lemental)-move process decomposes "elemental" moves into the "primitive" motion segments commanded to Prim. Elemental moves, while not precisely defined, consist primarily of robot motions which take place between grasping operations (MOV-TO-OBJ, MOV-OBJ-TO), grasping operations themselves in a generic sense (GRASP-OBJ, RELEASE), or the complete motion between source and destination locations when no object is involved (MOV-TO-LOC, MOV-TO-VU, MOV-TO-HOL, WITHDRAW). An elemental move commonly consists of a number of straight-line path segments.

The E-move level also decomposes some other functions which are more difficult to classify. For example, the E-move level interfaces with the vision system to determine the location of an object in a tray (LOCATE-OBJ). The E-move level decomposes the commands QC-DETACH and QC-ATTACH, as well, which perform the operations of moving into and releasing or attaching a gripper. There are four tool change commands which also get decomposed by E-move into straight-line motion primitives.

The E-move process sends commands to the Prim, Quick Change, and various gripper processes, thus coordinating robot motions with Quick Change and gripper actuation.

The routines for the E-move process are located at offset 18000, and the name of the E-move vocabulary is EDEF.

2.1. Commands

This section provides a brief description of the commands executed by E-move. In addition to the reference action variable, the routines executed by E-move commands depend on the status of commands sent to the gripper, Quick Change, and vision processes.

MOV-TO-LOC command

The MOV-TO-LOC command looks up the information for all the motion segments in a complete path, and sends the commands for the individual segments down to Prim. The information is retrieved from the data stored in common memory files (see Chapter VII), which includes such details as how to depart from the current location, what intermediate trajectory points to go through on the way to the destination, and how to approach the destination. No gripper commands result from execution of a MOV-TO-LOC command. E-move decomposes a MOV-TO-LOC command into a series of GO-THRU commands and one GOTO-PT command to Prim.

MOV-TO-OBJ command

The MOV-TO-OBJ command performs a similar function as the MOV-TO-LOC command, except that the "destination" is now an object at a location. Additional information is needed to determine the trajectory to and destination location for an object, since an object at a given location may be gripped in a variety of different ways, each of which requires a different robot location and approach trajectory. In RCS-II/T3, the different grips of an object at a location are indicated by "grip numbers". The grip number is given in the command to Task, and the position offsets and approach trajectories to use at each location have been

stored in the data associated with the object. MOV-TO-OBJ looks up this information in order to decompose the command into the proper primitives. MOV-TO-OBJ also commands the gripper to position the opening of the fingers as required to properly approach an object.

MOV-OBJ-TO command

The MOV-OBJ-TO command is used to move an object to a destination. This command functions very similary to the MOV-TO-OBJ command, except that, since the object is already grasped, the gripper does not need to be positioned for the approach to the destination.

MOV-TO-VU command

MOV-TO-VU is used to move to the picture-taking location if vision is required for an object. MOV-TO-VU is very similar to MOV-TO-LOC, except that different data need to be looked at for view points, and the gripper needs to be commanded to open fully (the camera looks down between the fingers) for all objects.

LOCATE-OBJ command

The LOCATE-OBJ command is used after the view point has been moved to and the exact object location is to be determined using the vision system. LOCATE-OBJ first commands the vision system to take a "fast flood" picture of the object to get an approximate location of the object centroid. After receiving the centroid coordinates from vision, the robot is commanded to position the camera directly above the center of the object. The vision system is then commanded to take a second picture, which will provide a more accurate indication of the object centroid, as well as the orientation angle of the object. The vision system also attempts to recognize the object based on this second image. If the object is not recognized, the vision system takes several pictures from the same position, varying system parameters in an attempt to make the image look like the desired object (see the AMRF vision system documentation for further details). The robot is then commanded to reposition above the part, and with the correct gripper orientation. When this command is finished, the location of the object is known, and a MOV-TO-OBJ command may be issued.

MOV-TO-HOL command

MOV-TO-HOL is used to move to a holster prior to a QC-ATTACH or QC-DETACH. It is similar to the other MOV-TO commands except that the gripper (if attached) is commanded to close to the value

associated with the holster object in the data (to prepare for moving down into the holster).

QC-DETACH command

The QC-DETACH command performs the necessary actions to move a gripper down into a holster and detach it from the robot. First, a GO-UNTIL command is sent to Prim, to be terminated upon the sensing of the gripper "ears" by the large holster proximity sensors. This occurs when the gripper ears are less than half an inch away from the holster. At this point, the Quick Change is unlocked, and the gripper is allowed to drop onto the holster. The mating Quick Change elements are still in contact, however. Next, a GO-UNTIL command is used to move the gripper forward in the holster to the seat position, a move which is terminated by the detection of the gripper ears by the small proximity sensors on the front of the holster. Finally, a DELTA-MOVE is sent to Prim to extract the robot up out of the gripper. QC-DETACH checks that the gripper has successfully detached.

QC-ATTACH command

QC-ATTACH performs the actions required to move the robot down onto a gripper, lock the gripper on with the Quick Change, slide back out of the seat position, and move the gripper up out of the holster. The first part of this command is accomplished with two DELTA-MOVES; one which moves the robot down close to the gripper fairly rapidly, and one which slowly mates the robot with the gripper. The command is then sent to the Quick Change to lock the gripper on. Following this, the gripper is commanded to close to make sure it may be pulled out of the holster. A DELTA-MOVE is then sent to Prim to slide the gripper back, followed by another to lift the gripper a distance out of the holster. At this point, the gripper id is checked to make sure the gripper is properly attached and is the correct one. Then another DELTA-MOVE is sent to move the gripper up completely out of the holster. Finally, the gripper which has just been attached is calibrated if required. This is only necessary for the rectangular parts gripper the first time it is attached.

E-PAUSE command

E-PAUSE is used to command the process to do nothing.

GRASP-OBJ command

The GRASP-OBJ command is sent between MOV-TO-OBJ and MOV-OBJ-TO commands to command the gripper to grasp the object. It results in E-move sending a grasp command to the appropriate gripper

control process.

RELEASE command

RELEASE is used to tell E-move to command the gripper to open and release the object. It results in E-move sending a release command to the appropriate gripper process.

WITHDRAW command

The WITHDRAW command may be used to execute a DELTA-MOVE from above the E-move level. This command is not currently used.

INSERT-TOOL command

The INSERT-TOOL command is used to insert a tool into the machine tool drum. E-move decomposes this command into the Prim commands DELTA-MOVE, CAL-SENSOR, and a GO-WHILE which monitors insertion force during the final motion.

RELEASE-TOOL command

RELEASE-TOOL is used after a tool has been inserted into the machine tool drum. First, E-move commands the tool gripper to open. Then, the force sensor is calibrated for the GO-UNTIL command which follows, sliding the gripper to the right while monitoring the moment about the tool Y axis. Finally, a DELTA-MOVE is commanded to Prim to pull the robot back away from the machine tool.

ACQUIRE-TOOL command

ACQUIRE-TOOL is used to move to and grasp a tool in the tool drum. E-move commands the tool gripper to open, then sends a DELTA-MOVE to Prim to move the robot toward the machine tool. The force sensor is then calibrated for a GO-UNTIL command which moves the robot into contact with the strike plate while monitoring force in the tool X direction. Another CALIBRATE-SENSOR command is then sent, to prepare for the GO-UNTIL command which slides the gripper to the left, into position to grasp the tool. Finally, E-move commands the toll gripper to grasp the tool.

REMOVE-TOOL command

Once a tool in the tool drum has been grasped, REMOVE-TOOL is used to pull the tool out of the drum. First, a CALIBRATE-SENSOR command is sent to Prim. This is followed by a GO-UNTIL command which moves the robot to pull the tool out of the drum while

monitoring the extraction force.

2.2. Interfaces

The E-move input command interface from Subtask consists of the following variables:

command-#-in
: integer; gets incremented every time a new command is sent.

"command-in"
: 10-character string; contains input command.

"obj"
: 16-character string; contains the name of the object.

obj-para-^
: integer; points to the record in the OBJ-PARA file which contains the parameters, such as those providing grip and vision information, for the object.

obj-grip-loc-^
: integer; points to the record in the OBJ-GRIPS file which contains the end effector id# and tray grip number to use for the object.

obj-vision-para-^
: integer; points to the record in the VISION-PARA file which contains the vision parameters for the object.

locpt-name
: 10-character string; name of the current location point goal.

locpt-type
: integer; indicates if the location point is an array or owner.

locpt-^-in
: integer; points to the record in the LOCPT file which contains the information about the current location point.

pose-^-in
: integer; points to the record in the POSE file which contains the current goal pose.

pose-type
: integer; indicates the type of the current pose.

endeff-id#-in
    integer; indicates the id number of the required gripper.

The output status interface to the Subtask level consists of the following variables:

command-#-echo-out
    integer; contains an echo of the input command number.

status-report
    12-character string; contains the output status. The possible values are:

Value	Description
executing	The current command is still executing.
done	The current command is finished.
error-state	An error has occurred during execution.
joystick-on	The Joystick Box is active.

status-para-out
    integer; status parameter available for more detailed status.

ref-pose-^
    integer; points to the record in the POSE file that contains the feedback pose to Subtask.

obj-held-status-out
    integer; flag which indicates if an object is held or not.

pres-endeff-id#-out
    integer; indicates which end effector is currently attached.

3. SUBTASK

The Subtask process decomposes output commands from Task into commands to E-move. For example, the Subtask input command GET-OBJECT gets decomposed into the E-move commands MOVE-TO-VIEW, LOCATE, MOVE-TO-OBJECT and GRASP-OBJECT. A primary purpose of the Subtask process is to coordinate the robot with the Active Pedestal to perform object reorientation. The Subtask process is located at offset 17000, and the Subtask vocabulary is called STDEF.

3.1. Commands

This section explains the purpose of the various commands executed by Subtask. In addition to the reference action variable, the execution of E-move commands depends on such variables as the end effector state (right, wrong, none), the pedestal status, and the object status (held, not held).

GET-ENDEFF command

The GET-ENDEFF command is used to connect the proper gripper to the robot. If the correct gripper is already on, then the command returns done immediately. Otherwise, a MOV-TO-HOL command is sent to E-move to prepare to remove the currently-attached gripper. A QC-DETACH is then issued, followed by a MOV-TO-HOL to move to the holster containing the desired gripper. Finally, a QC-ATTACH is commanded which attaches the desired gripper.

FIX-GRIP command

FIX-GRIP is used any time the object must be regripped by the robot with the aid of the the Active Pedestal. First, the pedestal is sent a command to move to the position to receive the object. Then, a MOV-OBJ-TO command is sent to E-move to move the object to where it can be grasped by the pedestal. The pedestal is then commanded to grip the object. Following this, a RELEASE and MOV-TO-LOC are sent to E-move. The pedestal is then commanded to reorient the object. Commands are then issued to E-move to move the robot to the object and grip it. This is followed by a command to the pedestal to release the object, which completes the FIX-GRIP command.

GET-OBJECT command

GET-OBJECT is used to move to an object and grasp it. If the object requires the use of vision, then a MOV-TO-VU is the first command sent to E-move. Then a LOCATE-OBJ is commanded to locate the object with the vision system and center the robot above the part with the proper orientation. This is followed by a MOV-TO-OBJ command and a GRASP-OBJ command, which complete the GET-OBJECT.

PLACE-OBJ command

The PLACE-OBJECT command is used to move an object to a location and release it. The E-move commands which are sent to accomplish this are MOV-OBJ-TO and RELEASE.

GOTO-ENDPT command and GOTO-DEST command

GOTO-ENDPT and GOTO-DEST are both used to move to a location. GOTO-DEST is used to move to the location specified in the destination field of the Task command, while GOTO-ENDPT is used to move to the end-at position specified by the Task command. The end-at position is a position moved to after the real purpose of a command has been accomplished. Both commands issue a MOV-TO-LOC to E-move.

MOVE-OBJ>D command

MOVE-OBJ>D is used to move an object to the destination location. It results in a MOV-OBJ-TO being sent to E-move.

S-PAUSE command

S-PAUSE is used to command the Subtask process to do nothing.

>POSITION command

>POSITION is used to move to a position, while using the parameters associated with objects. This command is used with virtual objects during "hold-and-seat" operations (see Chapter VII, section 7). Subtask sends a MOV-TO-OBJ command to E-move when it receives a >POSITION.

RELEASE command

RELEASE is used to open the gripper and release the currently-held object. It results in a RELEASE being sent to E-move.

INSERT-TOOL command

INSERT-TOOL is used to insert a tool in the machine tool drum. An INSERT-TOOL command is sent to E-move to accomplish this.

RELEASE-TOOL command

RELEASE-TOOL is used to release a tool once it has been inserted into the tool drum. A RELEASE-TOOL is sent to E-move to accomplish this.

ACQUIRE-TOOL command

ACQUIRE-TOOL is used to move to and grasp a tool in the tool drum. It results in an ACQUIRE-TOOL being sent to E-move.

REMOVE-TOOL command

REMOVE-TOOL is used to remove a tool from the tool drum once it

has been acquired. It results in a REMOVE-TOOL being sent to E-move.

3.2. Interfaces

The Subtask input command interface consists of the following variables:

command-#-in
integer; incremented every time a new command is sent.

"command-in"
10-character string; contains the command from Task.

"obj"
16-character string; contains the name of the object.

obj-endeff-id#
integer; contains the id number of the gripper used with the object.

src-grip-#-in
integer; contains the number which identifies the grip to use for the object at the source location.

dst-grip-#-in
integer; contains the number which identifies the grip to use for the object at the destination location.

obj-para-^
integer; points to the record in the OBJ-PARA file that contains information about the object.

locpt-name-in
10-character string; name of the current location point goal.

locpt-type-in
integer; indicates if the location point is an array or owner.

locpt-^-in
integer; points to the record in the LOCPT file which contains the information about the current location point.

pose-^-in
integer; points to the record in the POSE file which contains the current goal pose.

The output status to Task consists of the following variables:

command-#-echo
integer; contains an echo of the input command number.

status-report
12-character string; contains the status sent to Task. The possible values are:

Value	Description
executing	The current command is executing.
done	The current command is finished.
error-state	An error has occurred in executing the command.
joystick-on	The Joystick Box is active (robot is under manual control).

status-para-out
integer; available for additional status information, not currently used.

ref-pose-^-out
integer; points to the record in the POSE file that contains the feedback pose to Task.

obj-held-status-out
integer; indicates whether the gripper is grasping an object.

pres-endeff-id#-out
integer; contains the id number of the gripper currently attached to the robot.

4. TASK

The TASK process interprets and performs the first level of decomposition of robot task commands sent from the Horizontal Workstation Controller (HWSC), entered at the keyboard or loaded from a block. Task performs a number of file operations to retrieve the information needed to perform a command. The Task process also interfaces with the database, providing update information when objects are removed from or replaced in trays. The code for the Task process is located at offset 16000. The Task vocabulary is named TDEF.

4.1. Commands

This section explains the function of each of the Task input commands. The execution of Task-level commands depends on such variables as the state of the database (done, executing, etc.), the object status (held, not held), and a variable which indicates whether an object is to be handled. The Task level, of course, also has its own reference action variable.

TRANSFER command

TRANSFER is used to move an object from one location to another. To accomplish this, first the gripper is exchanged as required for the one needed to handle the part by issuing a GET-ENDEFF to Subtask. Then, the robot moves to acquire the object at the source location as a result of a GET-OBJECT command to Subtask. If the source grip number is different from the destination grip number, then the object will be moved to the Active Pedestal for regrasping. This is performed by commanding a FIX-GRIP. The object is next placed at the destination location and released via a PLACE-OBJECT command. Finally, a GOTO-ENDPT command is sent to move the robot to the end-at location (see 4.2) and the database is updated if the transfer was to or from the MBD.

MOVE command

MOVE may be used to move an object to a location and hold it there or to simply move the robot to a desired position. MOVE always results in the robot being positioned at the destination location; the end-at specification is ignored for this command. If no object is to be moved, a GOTO-DEST is simply sent to Subtask. If an object is to be moved, then first a GET-OBJECT is commanded, followed by a FIX-GRIP to reorient the object if needed. Finally, a MOV-OBJ>D command is sent to move the object to the destination and, again, the database is updated if the move was to or from an MBD.

POSITION command

The behavior resulting from the POSITION command depends on whether or not an object is being held. If an object is held, it is immediately released by sending a RELEASE to Subtask, and the robot then moves to the destination location as the result of Task sending a >POSITION command. If no object is held, then only the >POSITION is sent. A virtual object may be named in a POSITION command in order to specify a grip opening and position offset at the destination location. The virtual object is not a physical object which can be grasped or moved, but rather a convenience which enables the usual object mechanisms to be used

to determine grip opening, position offsets, and approach and depart trajectories for the command.

ACQUIRE command

The ACQUIRE command directs the robot to get the proper end effector for the object specified, move to the object and grasp the object. The commands sent to Subtask are GET-ENDEFF and GET-OBJECT.

REFIXTURE command

The REFIXTURE command is used when an object in the vise requires refixturing before machining can continue, and the refixturing requires the use of the Active Pedestal to reorient the object. If the object is not currently grasped, the REFIXTURE command first checks for the correct gripper to remove the part from the vise and performs a quick change as necessary, by sending a GET-ENDEFF command to Subtask. A GET-OBJECT is then sent to cause the robot to move to and grasp the object. In practice, the REFIXTURE command is almost always preceded by an ACQUIRE command, in which case Subtask immediately reports done for the GET-OBJECT command. Once the object is held, a FIX-GRIP command is sent to Subtask to move the robot and the Active Pedestal to reorient the object. Once the object has been grasped with the new orientation, Task issues a MOVE-OBJ>D to move the object to the destination (usually the programmable vise).

PAUSE command

The PAUSE command is used to command the robot to do nothing, and to move the robot around under manual control with the Joystick Box. PAUSE sends a PAUSE command down to Subtask, and it propogates down the hierarchy. PAUSE is also used during the robot startup sequence. When the Joystick Box is active, Task automatically sends down a PAUSE to cancel any current commands which may be running so that control is passed to the Joytick Box.

Four commands are used to enable tools to be changed on the HMC. These are INSERT-TOOL, RELEASE-TOOL, ACQUIRE-TOOL and REMOVE-TOOL. These commands were created to enable forces to be monitored during tool changing. They are used only for tool changing, however, and may not be executed with an arbitrary object at an arbitrary location. The tool changing procedure and commands are discussed in ref [12]. They are also summarized below.

INSERT-TOOL command

The INSERT-TOOL command is used to insert a tool into the tool drum after the tool has been picked up from a tray and moved to a position (TOOL-SAFE) near the tool drum with a MOVE command. The command checks for these prerequisites and then moves the tool into the transfer location while monitoring forces. An INSERT-TOOL command is sent to Subtask if the prerequisites for the command have been met.

RELEASE-TOOL command

RELEASE-TOOL is used to let go of a tool once it has been placed in the transfer position and the HMC fingers have gripped it. The command directs the robot to open the gripper fingers, perform a force-guarded slide along the strike plate near the tool drum and move to the end-at location. This is performed by sending a RELEASE-TOOL command to Subtask.

ACQUIRE-TOOL command

When issued an ACQUIRE-TOOL command, the robot first performs a quick-change if necessary to obtain the tool gripper, then moves to the location specified in the destination field (TOOL-SAFE). This is followed by a force-guarded move to the strike plate, a guarded slide along the strike plate toward the tool, and grasping of the tool. An ACQUIRE-TOOL command to Subtask accomplishes these actions.

REMOVE-TOOL command

REMOVE-TOOL is used to take the tool out of the tool drum after it has been acquired by the robot and released by the drum fingers. REMOVE-TOOL withdraws the tool from the tool drum, while monitoring forces, and moves to the end-at location (TOOL-SAFE). Thus is performed by sending a REMOVE-TOOL to Subtask.

There are two commands which are available for connecting and disconnecting the database system to RCS-II/T3. They are STARTUP and SHUTDOWN, and are described below. Specific details of the database protocol and startup and shutdown procedures may be found in the AMRF Data System documentation.

STARTUP

This command connects RCS-II/T3 to the database. As required, STARTUP first sends an Abort to the database, then an Initiate, and finally a Startup.

SHUTDOWN

SHUTDOWN is used to disconnect RCS-II/T3 from the database. The SHUTDOWN command sends an Abort to the database.

4.2. Interfaces

The input command interface to Task from the HWSC consists of the following:

cmdf<w
    4-character string; network protocol variable.

cmd-length<w
    integer; available to specify the length of the command, not currently used.

command-#-in
    integer; incremented every time a new command is sent.

time-stamp-status<w
    12-character string; available to provide a time stamp for the command, not currently used.

"command-16"
    16-character string; contains the Task input command.

"obj"
    16-character string; contains the name of the object to be handled.

"obj-serial-#"
    16-character string; contains the serial number of the object.

source-grip-#
    integer; contains a number that specifies the relative position and orientation of the gripper for the object at the source location.

"src-loc-16"
    16-character string; contains the location of the object at the start of the command.

src-sector#
    integer; if the source location is an array, indicates the sector which contains the part.

dest-grip-#
    integer; contains a number that specifies the relative position and orientation of the gripper for the object at

the destination location.

"dest-loc-16"
16-character string; contains the location to which an object is to be moved.

dest-sector#
integer; if the destination location is an array, indicates the sector in which to place the part.

"end-at-16"
16-character string; contains the name of the location to which the robot is to be moved at the end of the command.

endatsector#
integer; if the end-at location is an array, specifies which sector to move to.

The output status interface to the HWSC consists of the following variables:

fdbf>w
4-character string; network protocol variable, not currently used.

echo>w
16-character string; network protocol variable, not currently used.

status-length>w
integer; available to provide the length of the status message.

time-stamp-status>w
12-character string; available to provide a time stamp on the status.

"command-echo"
16-character string; contains an echo of the input command.

"status"
16-character string; contains the status sent to the HWSC. Possible values are:

Value	Description
executing	The command is executing.
finished	The command is finished.

VI. OTHER CONTROL SYSTEM PROCESSES

The processes that interface the RCS-II/T3 control system with the robot and control auxiliary equipment such as the quick change, grippers, control system graphics display, and active pedestal are described in this chapter. Again the general function, specific commands and communications interfaces of each process will be discussed.

1. T3 INTERFACE

The purpose of the T3 Interface (T3I) process is to transform the output of the RCS-II/T3 control system into the format required by the commercial T3 controller. The control system is interfaced to the T3 controller via a Dynamic External Path Control (DEPC) link (see Appendix A). The process also transforms feedback information from the resolver joint angle measurement system into the format required by RCS-II/T3. The third function performed by the T3I process is to monitor the DEPC communications for errors.

The DEPC link accepts commands giving the position of the robot end plate, in Cartesian/Euler format. The position commands may be sent at multiples of 10 ms (currently configured for 40 ms). The link also provides commands for restart and abort. Position commands are checked for velocity limits (the link aborts when a limit is exceeded). Communications between the T3I process and the DEPC link occur over a 9600 baud RS-232 serial line. The name of the T3I DEFINITIONS, or T3I. The T3I process is located at offset 19000 and is loaded on board 4.

1.1. Commands

This section describes the commands accepted by the T3I process.

RESTART command

The RESTART command is sent over the DEPC link to clear any errors and initialize the link. The T3I process waits for the feedback position, which is then sent as the command on subsequent cycles and transformed to frame format for feedback to RCS-II/T3.

POINT command

The point command is used to command all T3 motions. The goal frame from Prim is transformed to T3 format, and the resultant velocity is compared to the DEPC limit. If the limit is exceeded, the motion is scaled back to meet the limit. The

result is sent over the DEPC link, and also transformed back to RCS-II/T3 format for feedback to Prim.

PAUSE command

The last position output by the T3I process is repeated.

1.2. Interfaces

The input command interface from Prim consists of the following variables:

cycle-cnt-#-in
: integer; indicates the value of the cycle count when the command was written.

inc-command-#-in
: integer; incremented every time a new command is sent.

input-command
: 10-character string; contains the input command.

frame-command-in
: 9-element integer array; contains the values which define the commanded frame in RCS-II/T3 format (tool point coordinates, wrist to tool point vector, tool to finger point vector).

The output status interface to Prim consists of the following variables:

cycle-cnt-#-status-out
: integer; contains the cycle count when the status was written.

cycle-cnt-#-echo-out
: integer; echo of the cycle-cnt-#-in.

inc-command-#-echo
: integer; echo of the inc-command-#-in.

input-command-echo
: 10-character string; echo of the input command.

status-report
: integer; reports the overall status of the process. The following values are possible:

Value	Description
0	The command has finished (output position equals input position).
1	The command is executing.
2	An error has occurred.
3	The process is ready for a new point (which should be sent to avoid velocity discontinuity).

status-arg-out
integer; indicates the type of error which has occurred. The possible values are:

Value	Description
0	No error has occurred.
1	The DEPC link is not responding.
2	DEPC detected an error.
3	A message from DEPC contained a checksum error.
4	The message to DEPC was not transmitted.
5	An invalid command has been sent.

frame-feedback
9-element integer array; contains the frame commanded to the T3 in RCS-II/T3 format.

scale-factor-out
integer; contains a resolver scale factor.

joint-limit-dummy
integer; created to indicate when a joint limit is exceeded (not currently used).

2. QUICK-CHANGE

The Quick Change process interfaces the control system with the hardware used to control the Quick Change. The Quick Change process takes commands from E-move, which coordinates gripper activity. The functions performed by the QC-LEVEL are to lock and unlock the Quick Change, to check the QC sensors to make sure locks and unlocks have been successful, and to read in the value

of the end effector id number, which indicates which gripper is connected. The name of the Quick Change process vocabulary is QDEF. The Quick Change process is located at blocks 25100-25199 on the RCS-II/T3 disk, and is loaded on board 8.

2.1. Commands

QC-LOCK command

QC-LOCK locks the Quick Change device, and checks the quick change sensors to make sure the locking device did not get stuck. After the Quick Change is locked, the control port is disabled to prevent accidental unlocking of the gripper.

QC-UNLOCK command

QC-UNLOCK unlocks the Quick Change device, and again checks the Quick Change sensors to detect a Quick Change or sensor malfunction.

QC-PAUSE command

QC-PAUSE commands the Quick Change device to do nothing. The Quick Change and end effector id sensors are checked in order to provide the appropriate status to E-move.

2.2. Interfaces

The input command interface to the Quick Change process consists of the following variables:

command-#-in
integer; incremented every time a new command is sent.

"command"
10-character string; contains the input command from E-move.

The output status interface to E-move consists of the following variables:

command-#-echo
integer; echo of the input command number.

status-report
12-character string; contains status to E-move. The following values are possible:

Value	Description
executing	The command is executing.
qc-locked	The Quick Change is locked; there may or may not be an end effector present.
qc-unlocked	The Quick Change is unlocked; there may or may not be an end effector present.
qc-sens-malf	A malfunction of the Quick Change sensors has been detected.
qc-stuck	The Quick Change sensors indicate that the locking ring is either not fully locked or unlocked.

status-para-out
integer; available for additional status information; not currently used.

endeff-id#
eight-bit binary; indicates the id number of the gripper currently mated (though it may or may not be locked) with the Quick Change.

3. GRIP

The purpose of the Grip process is to control the actions of the rectangular parts gripper. The process performs position or force servoing, depending on the input command, of the gripper by controlling the eight valves which supply air to the gripper actuator. The name of the vocabulary for the Grip process is GDEF. The grip process may be found in blocks 23000-23999, and is loaded on board 8.

3.1. Commands

The Grip process is capable of executing the following commands:

POSITION command

The POSITION command is used to open the parts gripper to the desired opening. Any value between the minimum and maximum values may be specified, since the gripper can servo to a desired position. This command is used to position the gripper fingers properly when a part is approached or departed, and also when the vision system is used.

OPEN-GRIP command

OPEN-GRIP performs the same function as POSITION. The different command name is used when it makes more sense to call the desired behavior "open".

CLOSE-GRIP command

CLOSE-GRIP performs a force-servoed grasp, which is used to grasp objects with the desired force.

UP-INIT command

The UP-INIT command calibrates the position and force sensors on the parts gripper. The gripper closes and opens, taking sensor readings to set sensor scaling variables. This command automatically executes when the system is booted, if the parts gripper is attached at that time. Otherwise, it is executed the first time the parts gripper is attached.

PAUSE command

If a PAUSE command is received, the commands to the valves controlling the gripper stay the same as they were.

3.2. Interfaces

The input command interface to the Grip process consists of the following variables:

sincr-cmnd#
: integer; incremented every time a new command is sent.

scommand
: 10-character string; contains the input command.

sforce
: integer; contains the commanded value of force, in lb.

sposition
: integer; contains the commanded value of position, in inches.

svelocity
: integer; contains the commanded value of velocity.

send-effector
: integer; contains id number of the end effector which is currently being commanded.

The output status interface to E-move consists of the following variables:

incr-cmnd#-echo
: integer; contains an echo of the input command number.

status
: 12-character string; contains the status to E-move. Possible values are:

VALUE	Description
executing	The command is still executing.
done	The command is finished (desired values of force, position have been attained).
paused	The Grip process is waiting for both fingers to contact.
serror	An unattainable command has been sent.

initialized-?
: integer; contains a flag which indicates if the gripper has been calibrated.

pos
: integer; contains the actual gripper position.

l-force
: integer; contains the actual force experienced by the left finger.

r-force
: integer; contains the actual force experienced by the right finger.

vel
: integer; contains the calculated value of the actual velocity.

finger-hgt
: integer; contains the value of the "finger height"--the difference between the axial position of the fingers at the current grip opening and the axial position of the fingers when they are closed.

grip
    integer; contains the value of the command bits to the air valves.

4. V-GRIP

The V-grip process controls the gripper used for cylindrical parts, or v-gripper. This gripper is currently operated in a stricly open/close manner; there is no servoing of position or force. However, the position of the gripper fingers is monitored to determine the status of command execution. The V-grip vocabulary is named VDEF. The process is located at blocks 25200-25299 and is loaded on board 8.

4.1. Commands

The following commands are executed by the V-grip process:

VG-GRASP command

This command is used to close the v-gripper. The finger position is checked to provide the proper status to E-move. The status for this command reports if the fingers are closed, opened, or stuck.

VG-RELEASE command

This command is used to open the v-gripper. The finger position is again checked to provide the proper status.

VGRIP-POSE command

This command also opens the gripper; the same routines as for VG-RELEASE are executed.

VG-PAUSE command

A PAUSE commands the process to do nothing. The position potentiometer is read to provide status. An additional status possibilty for PAUSE as to report that the fingers are at the commanded position.

4.2. Interfaces

The input command interface to the V-grip process consists of the following variables:

command-#-in
    integer; incremented every time a new command is sent.

"command"
: 10-character string; contains the command from E-move.

commanded-force
: integer; available force specification of grip force, not currently used.

commanded-size
: integer; contains commanded finger opening in inches.

commanded-vel
: integer; available for specification of velocity, not currently used.

endeff-id
: integer; contains id number of the currently-commanded gripper.

The output status to E-move consists of the following variables:

command-#-echo
: integer; echo of input command number.

status-report
: 12-character string; contains status reported to E-move. The following values are possible:

Value	Description
executing	The command is executing (gripper fingers are moving).
vgrip-opened	The fingers have stopped moving, and are either opened (VG-GRASP, VG-RELEASE, VGRIP-POSE, and VG-PAUSE) or at the commanded position (VG-RELEASE only).
vgrip-closed	The fingers have stopped moving, and are either closed (VG-GRASP, VG-RELEASE, VGRIP-POSE, and VG-PAUSE) or at the commanded position (VG-GRASP only).
jaws-stuck	The fingers have stopped, and are not at any of the closed, open, or commanded positions (all commands).
at-position	The fingers have stopped, and are at the commanded position (VG-PAUSE only).

status-para-out
integer; available to provide additional status information to E-move, not currently used.

vgrip-size
vgrip-left-force
vgrip-right-force
vgrip-vel
integers; these variables are available to report the sensed values of gripper finger position, force, and velocity. They are not used currently.

5. TOOL-GRIP

The process which controls the tool gripper, TG-LEVEL, is essentially identical to the VG-LEVEL process except that some of the gripper parameters are different, and "VG" is replaced with "TG". The commands and interfaces will therefore not be repeated here. The vocabulary name for the tool gripper process is TGDEF. TGLEVEL is loaded on board 8 and is located in blocks 300-399 at offset 25000.

6. DIAGNOSTICS AND GRAPHICS

The purpose of the Diagnostic and Graphics Processes is to provide the user with an up-to-date view of the states the system. The combined action of these two processes results in a graphics display of system processes and their current states, by which the user is made aware of the commands currently being executed by processes and the statuses returned.

The display shows the currently active gripper process, the status of data system updates, the status of Active Pedestal commands, the status of vision system commands, and the state of each process in the main RCS hierarchy, i.e. Task, Subtask, E-move, Prim, and T3 Interface. This detailed information describes the robot's current activities in real time, as the display is updated every other execution cycle of the system, i.e. every 80 milliseconds. When there are problems, the display can alert the user to the process which is having difficulty.

The following two sections will relate the details of the operation of these processes.

6.1 Diagnostics Process

The Diagnostics process reads the states of the system processes every execution cycle. This data is translated into appropriate formats for display. Then it is shipped to the Graphics process

via a large memory buffer, where it will be used to update the graphics display.

The Diagnostics process executes from board 0, and is one of only two processes residing on the board, (the other being Communications.) The source code for board 0 processes is located from blocks 24000 through 25000 on the system disk. Block 10 at 24000 OFFSET gives the overall directory of board 0 code. Sub-directories, containing more detail, can be found at each load block of block 10.

The Diagnostics Level is defined at blocks 901-909. This is the main routine of the process, called by the board executor SSGO. The board executor is defined at block 992. There are two principal components of the Diagnostics Level.

One principal component of the level is the reading of states of the system. The states of the system are generally the commands and statuses of the interfaces between processes. This information is resident within the other processes of the system. Therefore, Diagnostics must somehow access variables within other processes. This is done by defining data transfer variables that contain the addresses into the state variables of other processes. Block 120 contains these definitions. With these definitions, the states of the system can be read easily using a normal segv transfer. See blocks 220-229.

The only other component of real interest in Diagnostics is the graphics transfer routines. The transfer buffers are defined at block 180. These use pre-defined buffer addresses that are hard coded in the system. The actual transfers are defined at block 690. It is important to note that two separate buffers are used for data transfer. Diagnostics must read the address of the correct buffer from ascii-command-var each cycle. The Graphics process toggles between the two buffer addresses so that it always has a previous version of the data for comparison.

Changing or adding components of the diagnostic display is relatively simple. The user need only modify the transfer-variable definitions and segv transfers to acquire the new information. Then update the Graphics process to read this information from its buffer and display it. The Graphics process is described in the next section.

6.2 Graphics Process

The Graphics process receives state information from the Diagnostics process, and displays it on a color screen. In general, the state information displayed includes the current

command number, command name, status number, and status name for each process. The processes displayed include Task, Subtask, E-move, Prim, T3 Interface, the three gripper processes, Quick Change, Active Pedestal, Active Pedestal gripper, and Active Pedestal table. Also displayed are boxes representing the Workstation Controller, Data System, and vision system. The information displayed for these boxes represents only commands issued and status received by RCS.

The Graphics board, board 12, executes only the Graphics process each execution cycle. The Graphics data is displayed on a Vectrix graphics system through the serial port on the Graphics board.

The code to display the system diagnostics is refered to as the ASCII Display Screen (ADS). The code for ADS is at 27700 OFFSET on the system disk. This software draws the boxes, lines, and other icons in the display. It also displays the ASCII data of commands and statuses.

Each process for which command and status is to be displayed has its graphic entity defined in blocks 100-130. With each process is associated an INDEX in blocks 113-115. This INDEX points into the diagnostics data buffer. Thus, it indicates the address of the data for a particular process. Note that this INDEX must correspond to the order of the definition in Diagnostics, i.e. the variable owners in blocks 24110-24119. Any modifications to Diagnostics must be accompanied by appropriate modifications to the graphics entities defined at 27800 to achieve the desired result on the display.

7. ACTIVE PEDESTAL

The Active Pedestal (AP) process controls the actuation of the Active Pedestal device, which is used to grasp and reorient objects. The AP process receives commands from the Subtask process over the factory network, and controls both the hand and the rotary table to provide the proper response. The hand and table controller each have their own subprocesses, but these are transparent to the RCS-II/T3 controller. The AP process resides at offset 24000 on the AP system disk and is loaded on processor boards which are located in the pedestal controller bucket.

7.1. Commands

The AP process executes the commands given below.

PRE-GRIP command

PRE-GRIP is used to position the hand and the table prior to accepting an object from the robot. The gripper is opened to the commanded position and the table rotates to the specified (absolute) angle.

GRIP command

The GRIP command is issued after the robot has positioned the object within the grasp of the AP hand. The AP hand closes on the object with a force-servoed grasp. No command is sent to the rotary table.

FLIP command

FLIP is used to reorient the object once the robot has released the part and moved away. The table is rotated by the relative angle specified in the command parameter list. No command is sent to the hand.

RELEASE command

The RELEASE command is sent to direct the AP hand to open, and release the currently-held object. The hand opens to the commanded position. No command is sent to the rotary table.

PAUSE command

A PAUSE command directs the AP controller to do nothing.

7.2. Interfaces

The input command interface to the AP process consists of the following variables:

cycle-cnt
    integer; contains the value of the cycle count when the most recent command was sent.

incr-cmnd#
    integer; incremented every time a new command is sent.

command
    10-character string; contains the command from the Subtask process.

grasp-size
    integer; contains the commanded gripper opening in inches.

grasp-force
    integer; contains the commanded grasp force in lb.

table-angle
    integer; contains either the absolute angle to which the table is to be rotated (PRE-GRIP) or the angle relative to the current position by which the table is to be rotated (FLIP). The angle is specified in tenths of a degree in either case.

The output status reported to the Subtask process of RCS-II/T3 consists of the following variables:

cycle-cnt-echo
    integer; echo of the cycle count.

incr-cmnd#-echo
    integer; echo of the input command number.

status
    10-character string; contains the status reported to Subtask. The following values are possible:

Value	Description
executing	The command is executing; either the hand or table or both are still moving.
done	The command is finished; the table has reached the commanded position and the hand has achieved the commanded position or force.
can't-do	A command has been sent which cannot be successfully executed.
error	An error has been encountered which prevents the command from finishing.

status-code
    integer; available for additional status information, not currently used.

grip-size
    integer; contains the sensed value of the gripper opening in inches.

grip-force
    integer; contains the sensed value of the grasp force in lb.

table-position
    integer; contains the sensed value of the absolute table angle, in tenths of a degree.

VII. TASK DATA

This chapter describes the various data structures which are accessed by the control procedures to determine the specific behavior of the robot during command execution. The available structures provide for the specification of robot poses, locations (distinction to be made later), pose and location offsets, trajectories between locations, and object data such as grasp sizes and forces, parameters for the Active Pedestal and vision, approach and depart trajectories, and grip offsets at different locations.

Most data items have names so they may be referred to symbolically. All names except those for objects must be ten characters or less. Object names can be up to 16 characters.

The task data reside in a number of files, one of which has been created for each type of data. New entries are added to the files through the use of special data "forms". The forms are blocks that you edit to enter the desired information. They contain compiling words which take care of the system-level details of adding the data to the appropriate file when the block is loaded. The file declarations are located at offset 21000, blocks 100-199. The record structure for each file may be found here. The file declarations include a specification of the allowable number of records in the file, which may need to be increased by the user if a FILE FULL message is generated when new data is loaded. SMACRO files and file operations are discussed in chapter 7 of reference [1].

1. POSES

The data structure used in RCS-II/T3 to define a robot position and orientation in space is called a "pose". Pose forms are located at offset 21000, blocks 600-699. A typical pose form is shown in Figure VII.1. An entry in a pose form consists of nine values; three sets of Cartesian coordinates which define the position of a point located along the axis of the end effector (wrist point), the point at the center of the distal end of the end effector (tool point), and a point located along a radial vector out from the tool point (finger point). Figure VII.2 shows the three points and the tool coordinate system they define. The nine values for the robot at its current position may be displayed by pressing the record button on the Joystick Box (see section 3.3 of Chapter III). The units of poses are hundredths of an inch, and they are measured with respect to the robot world coordinate system. Poses are stored as records of the file named POSE.

```
BLOCK# 619
( POSE FORM
  x-val     y-val     z-val               "mov-name"  "pose-name"
|---------|---------|---------|---------|-----------|----------)

  6497     -2366     1569
  9495     -2475     1595
  9523     -1460    -1228      -pose-     COR-TL-TR   TOOL-TRNS
```

Figure VII.1. A Pose Form

Besides the three sets of coordinates, a pose form entry also contains the compiling word -pose-, the name of the pose, and the name of the "movetable", or auxiliary transformation, to be applied to the pose defined by the three points. Movetables, may be used, for instance, to apply small offsets to correct a pose. In Figure VII.1, the movetable COR-TL-TR is applied to correct the pose named TOOL-TRNS. Movetables are discussed further in the following section.

The pose definitions in blocks 601-609 are for poses which are used by the system. For example, the three poses T-POSE-1, T-POSE-2, and T-POSE-3 are used to store the actual values of the poses commanded to the Subtask process from Task. As mentioned in Chapter II, only a pointer value which indicates the record number of the pose to look at is then passed in the communications interface.

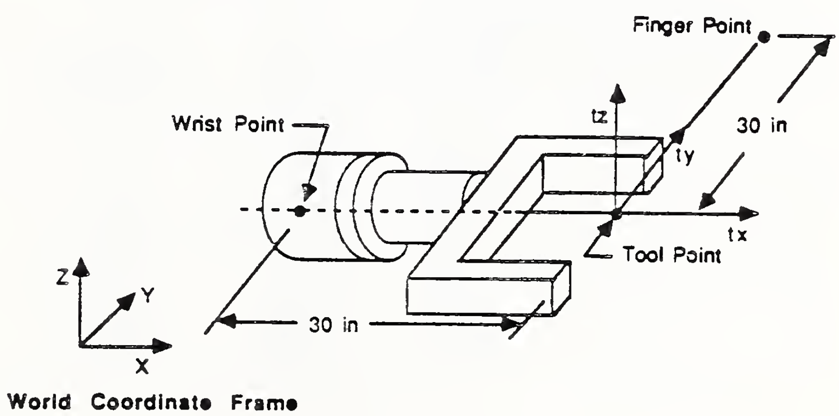


Figure VII.2. Wrist, Tool, and Finger Points for Poses

2. MOVETABLES

"Movetables" are general-purpose transformations available in RCS-II/T3 to define relative positions. They are used to modify poses, as seen above, to define "location points" (section 3), to define positions in arrays (section 4), to define approach and depart trajectories (section 7), and to set up goal trajectory points for some Prim-level commands such as DELTA-MOVE, GO-UNTIL, and GO-WHILE (Chapter V, section 1). Movetable forms are located at offset 21000, blocks 401-599. The movetable file is called MOVTAB.

A typical movetable is shown in Figure VII.3. Each line in a movetable specifies a rotational or translational offset in a particular coordinate system. The available coordinate systems are given in Table VII.1.

Movetables may consist of up to eight lines of translations and rotations which are applied additively in the order in which they are listed in the form. The units for translations is hundredths of an inch, and hundredths of a degree for rotations. Each line in a movetable form must be consecutively numbered beginning with 1, and all but the last line end with the compiling word ---. The last line in a movetable uses the compiling word -mtb-, and ends with the name of the movetable. There is a "null" movetable named ... which is used in forms requiring a movetable when no offset is to be applied.

BLOCK# 412

```
ERASE-MOVTAB   ( MOVTAB FORM                                    x
---------------                                               y-|
             | r,t  |  x   |  y   |  z  |                     |       |
             | to,wo| pit  | yaw  |     |                     |-------|
             | ro   | roll |      |     |                         |
      |------|------|------|------|-----|     ---                 |
      |line# | type | del  | del  | del |    -mtb-      "movtab-name"
      |------|------|------|------|-----|--------------|------------)
       1       t        0        0   -200      ---
       2       ro     -75        0      0      ---
       3       t        0        0    200      ---
       4       to     -60        0      0      ---
       5       t      -11      -10      6      ---
       7       ro      60        0      0     -mtb-    COR-TL-TR
```

Figure VII.3. A Movetable Form

Table VII.1. Movetable Coordinate Systems

	Specifier	Parameters	Effect
World:	rw	x, y, z	Translate pose by x, y, and z in world coordinates.
Tool:	t	x, y, z	Translate pose by x, y, and z in tool coordinates.
Tool Origin:	to	pit, yaw	Rotate pose by pit and yaw about the tool point.
Wrist Origin:	wo	pit, yaw	Rotate pose by pit and yaw about the wrist point.
Roll:	ro	roll	Rotate pose by roll about the end effector axis.

3. LOCATION POINTS

"Location points" are the locations used in commands to specify where the robot is to go. Location point forms may be found in blocks 720-729 at offset 21000. Figure VII.4 shows a location point form with a number of entries. An entry in the location point form consists simply of the compiling word -pt-, the

BLOCK# 721

```
( -pt-        "loc-name"      "pose-name"     "movtab-name"
 ------------|---------------|---------------|----------------)
  -pt-          HOME            HOME-POS        ...
  -pt-          UNIT_2          UT-2-P          ...
  -pt-          UT-2-TEST       UT-2-P          UT-TEST
  -pt-          UNIT_1          UT-1-P          ...
  -pt-          UT-1-TEST       UT-1-P          UT-TEST
  -pt-          PFIXTURE        PFIX-P          ...
  -pt-          H-VFIXTURE      HVFIX-P         ...
  -pt-          V-VFIXTURE      VVFIX-P         ...
  -pt-          PEDESTAL        PED-P           ...
  -pt-          FIX-1-SAFE      PFIX-P          FIX-1-SAFE
  -pt-          PED-SAFE        PED-P           PED-SAFE
  -pt-          SAFE            SAFE-P          ...
  -pt-          TABLE           TABLE-P         ...
  -pt-          UNIT-SAFE       UT-2-P          UNIT-SAFE
```

Figure VII.4. A Location Point Form

location point name, a pose name and a movetable name. The position and orientation of the location point are the result of the movetable applied to the pose. As seen in Figure VII.4, the location point form allows different command locations to be created from the same pose. When the locaton point is to be at the same position as the base pose, the null movetable is used. Location points are records in the LOCPT file.

4. ARRAYS

It is often desirable to be able to refer to locations in an ordered set by a single name and an index. This is convenient, for instance, when a part presentation tray is divided into a two-dimensional array of a number of "sectors". The "array" structure in RCS-II/T3 provides the desired capability. A number of arrays can be identified with a single location, such as an MBD unit, by using the "owner form," discussed in section 5. This allows different tray configurations to be used at a single location. Objects are associated with a particular type of array through the "object location form" (section 8). The location of a particular type of tray can be specified simply by the location of the tray (an owner of several different arrays) and the sector number. Sectors are numbered from left to right, top to bottom, from the point of view of the robot.

Two separate forms are required to define an array. Figure VII.5 shows three entries in an example of the first, the "array dimension" form. This form defines the name of the array, the number of dimensions of the array (1 or 2) and the number of elements in each dimension. The array dimension form also includes the name of the movetable to be used to calculate the

BLOCK# 702

```
( ARRAY-dim FORM
        |              | #  | #  | #  |     1st     |     2nd     | #
        |"array-name"  | dim| 1st| 2nd|  "movtab"   |  "movtab"   |pres
        |--------------|----|----|----|-------------|-------------|----)

 -arr-   SECTORS         2    2    2     -1050ty      -1050tz      0

 -arr-   TL-SECTORS      1    3           -800ty                   0

 -arr-   TRNG-SCTRS      2    2    4      -840tz        500ty      0
```

Figure VII.5. An Array Dimension Form

offset between sectors in each dimension. The compiling word for entries in array dimension forms is -arr-. Blocks 701-709 at offset 21000 are reserved for array dimension forms. The three entries in the array dimension form shown in Figure VII.5 are the only arrays currently defined. They are used for the three different types of trays; the one-dimensional, three-sector tool trays, the two-dimensional, four-sector trays used for prismatic parts, and the two-dimensional, eight-sector cylindrical part trays.

The other form needed to define an array is the "array location" form. Figure VII.6 shows the array location form for the three current arrays. Array location form entries are compiled by the word -ar-. When an array is to be used with an owner, the array location form identifies the movetable which specifies the offset from the "owner" location point to the location of the first sector in the array. In this case, the array location name (such as SECTORS, TL-SECTORS, TRNG-SECTORS) is associated with the owner, and is not used directly in commands. Instead, the owner name is used.

The array location form may also be used to define a simple static array at a particular location. To do this, the location name to be referred to in commands is given as the location name, the base pose for the array is specified in the "pose-name" area, and movetable name specifies the offset of the first sector from the base pose. Array location forms are located in blocks 710-719. Arrays are stored in a file named "ARRAY-TABLE."

BLOCK# 711

```
( -ar-       "loc-name"      "pose-name"    "movtab-name"
-----------|---------------|--------------|----------------)

 \ SECTORS is array to be accessed at UNIT_X + SECT-MTB

  -ar-        SECTORS          ...          SCT-MTB

  -ar-       TL-SECTORS        ...          TL-SCT-MTB

  -ar-       TRNG-SCTRS        ...          TRNG-SMTB
```

Figure VII.6. An Array Location Form

5. OWNERS

An "owner" is a structure which allows a number of locations or locations and arrays to be grouped together. One motivation for this capability, as seen earlier, is to be able to use different arrays at a single location (with the type of array depending on the type of object). Another instance where owners are useful is in defining common approach/depart or intermediate trajectories (see section 6) for a number of locations. If a number of locations are listed under an owner and trajectories are defined using the owner name, then the same trajectories are used for all locations listed under the owner. Otherwise, separate trajectories would have to be defined for each location.

Figure VII.7 shows an owner form containing owners which have been created to provide such convenience of trajectory definition. The entries in the owner form consist of the compiling word -own-, the name of the owner; and the compiling word -m-, and the type and name of each member. As a result of this owner, the location points UNIT_1 and UNIT_2 can both use trajectories defined for the owner UNIT.

The owner form used to create the owners for the different arrays to be used at a single location is shown in Figure VII.8. The elements of the owner definition are the same as above, except that the first member should be the location point at which the array members will be used. Owner forms are located at offset 21000, blocks 740-749, and the owner file is named OWNER-ASCII-NAMES.

BLOCK# 741

```
( OWNER FORM      mem-type =>  arr pos  lpt
                 "owner-name"          | mem-type | "mem-name"
                |--------------|-------|----------|------------)

   -own-          UNIT          -m-       lpt       UNIT_2
                                -m-       lpt       UNIT_1

   -own-          VFIXTURE      -m-       lpt       H-VFIXTURE
                                -m-       lpt       V-VFIXTURE
```

Figure VII.7. An Owner Form

BLOCK# 742

```
( OWNER FORM      mem-type =>  arr pos  lpt
                "owner-name"          | mem-type | "mem-name"
               |---------------|-------|----------|------------)
   -own-          UNIT_1        -m-        lpt       UNIT 1
                                -m-        arr       SECTORS
                                -m-        arr       TL-SECTORS
                                -m-        arr       TRNG-SCTRS
      \ Tray areas should include the LOCPT of the tray corner
      \ as the first member...the arrays are then offsets from
      \ this base pose to the various tray configurations

   -own-          UNIT_2        -m-        lpt       UNIT 2
                                -m-        arr       SECTORS
                                -m-        arr       TL-SECTORS
                                -m-        arr       TRNG-SCTRS

```

Figure VII.8. The Owner Form for the MBD Unit Arrays

6. INTERMEDIATE TRAJECTORIES

The "trajectory" forms provide a means of specifying intermediate "via" points and trajectory parameters between commanded locations. Figure VII.9 shows a block containing a typical trajectory form with two entries. Trajectory forms are compiled by the words -traj and -ppt. The word -traj is used to identify the name of the trajectory, as well as the types and names of the start and end of the trajectory. The possible types of start and end points of trajectories are array, location point, or owner.

The compiling word -ppt is used for the subsequent lines in a trajectory form entry, which define the points and parameters to use for each trajectory segment. In each -ppt line, the first parameter, acc-fac, specifies the maximum velocity, smooth-acc specifies a smoothing parameter, and brk-dst determines how close to a goal point the end effector must come before the robot will begin moving to the next point in the trajectory. The brk-dst is specified in hundredths of an inch. Increasing the smoothing parameter results in less smoothing and vice-versa. The fifth and sixth parameters, specify the type and name of the specific trajectory point or offset from the previous reference pose. A trajectory point may be a pose, location point, or movetable. The forms for intermediate trajectories may be found in blocks 850-889 of offset 21000. The files associated with trajectories are named TRAJ, TRAJ-PARA, and PATH-PTS.

```
BLOCK# 852
( -traj "traj-name" start-type "start-name" end-type "end-name"
          | acc | vel | smoo |  brk- |  ppt  |   path-pt
    -ppt  | fac | max | acc  |  dist | type  |  "name"
----------|-----|-----|------|-------|-------|-----------------)

-traj  HOME-SAFE       lpt  HOME       lpt  SAFE
    -ppt     4    100     8    300      pos       HOM-SAF-1
    -ppt     4    100     8    300      pos       HOM-SAF-2
    -ppt     4    100     8    400      pos       HOM-SAF-3

-traj  SAFE-HOME       lpt  SAFE       lpt  HOME
    -ppt     4    100     8    300      pos       HOM-SAF-3
    -ppt     4    100     8    300      pos       HOM-SAF-2
    -ppt     4    100     8    400      pos       HOM-SAF-1
```

Figure VII.9. A Trajectory Form for an Intermediate Trajectory

7. OBJECTS

Much of the data in RCS-II/T3 is associated with specific objects. The data which must be associated with an object includes the object name, grasping and vision information, and position offsets and approach and depart trajectories to use for different grips of the object at different locations. The above information must also be provided for partially machined versions, if any, of each part, as well as for "virtual" objects which are needed for the part. Virtual objects are used when it is desired to execute a command normally associated with an object, but no object is to be physically manipulated. An example of the use of virtual objects is in the "hold and seat" procedure described in Chapter VIII. The entire 1000 block area at offset 20000 is reserved for object data. The object data is organized by object, with 25 blocks for each. Block 0 provides a directory to the data for the objects. The examples used in this section are taken from the data created for a part called a "locking clevis".

7.1. Object Form

The object form is used to name an object and to associate grasp parameters with it. An example object form is shown in Figure VII.10, with four entries. This form names the different objects

BLOCK# 453

```
( OBJECT FORM
                            ref   appr  grasp  grasp  depart
              "obj-name"   grip#  size  size   force  size
         |================|=====|=====||=====||=====||======)

 -obj-    LCLEVIS_BH         2    500   250     35    370
\ pick up from tray

 -obj-    LCLEVIS_IH         1    370   250     50    400
 -obj-    LCLEVIS_IH        -1    500   250     40    370

 -obj-    LCLEVIS            1    370   250     40    400
\ remove from fixture, place on tray
```

Figure VII.10. An Object Form

needed to make the locking clevis. Names for objects which enter and leave the workstation are defined by higher-level systems in the AMRF, since they must be consistent throughout the factory. LCLEVIS_BH is the name of the object which is the blank for the locking clevis part. LCLEVIS_IH is the name of a partially-machined intermediate locking clevis part, and LCLEVIS is the name of the finished part.

Entries in an object form consist of the compiling word -obj-, the object name, a "reference grip number," approach, depart, and grasp sizes for the object, and the grasp force to be used when gripping the object. The reference grip number is a number that indicates the position offset to be used when gripping the object. Grip numbers are discussed further in section 7.3. A separate object form entry must be made for each different grip number of an object, since different grips will usually require different grasp parameters. An entry in an object form creates a new record in the OBJ-ASCII-NAMES and END-EFF-PARA files.

7.2. Object Grips File

The file named OBJ-GRIPS associates an object with the gripper that is to be used to handle the object. It also identifies the grip number used when accessing a part from a tray. There is at present no compiling word to add records to the OBJ-GRIPS file, but this can be easily accomplished by using the format used in the block shown in Figure VII.11. This block makes use of the

```
BLOCK# 455
FDEF
:S
        "object"     <=  LCLEVIS_BH
        endeff-id#   <=  1
        tray-grip#   <=  2
:R      >FILE   OBJ-GRIPS

:S      "object"     <=  LCLEVIS_IH
        endeff-id#   <=  1
        tray-grip#   <=  1
:R      >FILE   OBJ-GRIPS

:S      "object"     <=  LCLEVIS
        endeff-id#   <=  1
        tray-grip#   <=  1
:R      >FILE   OBJ-GRIPS
```

Figure VII.11. A Block which Adds a Record to the OBJ-GRIPS File

Show mode to set the values of variables in an OBJ-GRIPS record. The word >FILE adds the record to the file when the block is loaded. It is seen that the prismatic parts gripper (endeff-id# = 1) is to be used for all stages of the locking clevis part. The tray grip number is 2, for the part blank, and 1 for the other parts.

7.3. Object Location Form

The object location form is used to associate an object with the particualar member array to be used for the object at the MBD units. For example, in Figure VII.12, it is seen that both

```
BLOCK# 456
( OBJ-Loc  FORM       assigned-type => lpt arr
-----------------


                                   assigned        assigned
              "object-name"        loc-type        "loc-name"
             |---------------|------------|---------------)

-o-loc-      LCLEVIS_BH              arr          SECTORS

-o-loc-      LCLEVIS                 arr          SECTORS
```

Figure VII.12. An Object Location Form

objects LCLEVIS_BH and LCLEVIS use the SECTORS array (see section 4). The intermediate object LCLEVIS _ IH does not appear in an object location form entry because it is never placed in or removed from a tray. Object location form entries must have the compiling word -o-loc-, the name of the object, the assigned location type, and the assigned location name. Owner information is stored in a file named OWNER-ASCII-NAMES.

7.4. Pedestal Parameter File

When an object is to be reoriented using the Active Pedestal, a record for the object must be added to the file named PED-PARA. As with the OBJ-GRIPS file, there is not currently a compiling word to add records to PED-PARA. Records are added to PED-PARA using the same mechanism as for OBJ-GRIPS. Figure VII.13 illustrates the variables which need to be set for a PED-PARA record. First is the name of the object to be handled by the AP, then come the grasp parameters to be used by the AP in gripping the object. Note that the grasp parameters are similar to those defined in the object form. A PED-PARA record also includes the angle which describes the initial position of the rotary table when the object is first grasped by the AP and the angle of table rotation when the part is flipped around. A ped-table-pos of -235 positions the table so the AP fingers open and close perpendicularly to an imaginary radial line from the robot center. A ped-flip-angle of 1835 flips the part 180 degrees. Since the LCLEVIS_IH intermediate object is the only one for this part that gets handled by the AP, there is no need for the others to have records in PED-PARA.

BLOCK# 457

```
FDEF

:S       "p-object"          <=  LCLEVIS_IH
         ped-appr            <=  500
         ped-grasp-size      <=  250
         ped-force           <=   50
         ped-release-size    <=  450
         ped-table-pos       <=  -2135
         ped-flip-angle      <=  1835
:R     >FILE   PED-PARA
```

Figure VII.13. A Block which Adds a Record to the PED-PARA file

7.5. Grip Location Point/Movetable Form

The grip location point/movetable form assigns the movetables to be used for different grips of an object at different locations. A typical grip location point/movetable form is shown in Figure VII.14. Entries in the form are compiled by the word -grip-. The other parameters in a grip location point/movetable form entry are the object name, the grip number, the location name and type, and the movetable to be used at that location for that object with that grip number. The indicated movetable is applied to the location point given when there is a command to do something with the object at that location with that grip number. Different grips of an object at the same location require different grip numbers and movetables to be defined and used, as in the third and fourth entry in the form in Figure VII.14. The wild card symbol $$$ is used to denote the movetable to use for a particular grip of an object at any other locations not specifically listed.

Negative grip numbers are used (for prismatic parts) to indicate that the grip is to be from the opposite side of the part as the grip with the same number, only positive. If a command to Task specifies that the source and destination grip numbers are opposite, this is an indication that the part must be flipped by the AP. For cylindrical parts, positive grip numbers are usually used for part blanks, and negative numbers for the finished part. See the following section for grip numbering conventions when using the vision system. Grip location point/movetable form entries create a record in the GRIP-LOC file.

BLOCK# 460

```
( GRIP-LOCPT-movtab FORM      ref-type => lpt arr $$$
                            ref   ref       ref          grip
             "obj-name"    grip#  type   "loc-name"    "mov-name"
------|------------------|---|------|------------|------------)

-grip-  LCLEVIS_BH         2    $$$                   LCL-GP+1
-grip-  LCLEVIS¯BH         2    lpt     PFIXTURE      LCL-GP+1F
-grip-  LCLEVIS¯IH         1    lpt     PFIXTURE      LCLI-GP+1F
-grip-  LCLEVIS¯IH        -1    lpt     PFIXTURE      LCLI-GP-1F
-grip-  LCLEVIS¯IH         1    lpt     PEDESTAL      LCLI-GP+1P
-grip-  LCLEVIS¯IH        -1    lpt     PEDESTAL      LCLI-GP-1P
-grip-  LCLEVIS¯IH        -1    $$$                   LCLI-GP-1P
-grip-  LCLEVIS¯           1    lpt     PFIXTURE      LCLF-GP+1F
-grip-  LCLEVIS            1    $$$                   LCLF-GP+1
```

Figure VII.14. A Grip Location Point/Movetable Form

7.6. View Location Point/Movetable Form

The view location point/movetable form is used to enter the information needed to locate the part with the vision system. The view location point/movetable form for the locking clevis blank is shown in Figure VII.15. There are two entries in the form, since there are two locations, UNIT_1 and UNIT_2, where the blank may be presented. Entries in a view location point/movetable form are compiled by the word -vu-. The object name and reference grip number are given for each entry. For the vision system, it is important to use grip numbers according to the following convention: grip 1 is across the widest dimension and grip 2 is across the shortest dimension. This information is used by the vision system to determine the orientation angle to provide to RCS-II/T3.

An entry in the view location point/movetable form also includes an ID number which identifies the part for the vision system, so the correct data will be used for part recognition. The vision ID numbers for all the parts currently made are listed at the bottom of the block shown in Figure VII.15. The location name and type where vision is to be used for the part is also specified. The "mov-name" specification names the movetable to be applied at the location to position the robot for taking the first picture of the part with the wrist-mounted camera. An expected range (in mm) is also included to help the vision system recognize the part. Vision information is stored in the VISION-PARA file.

BLOCK# 462

```
( VIEW-LOCPT-movtab FORM        v-ref-type => lpt arr
                                mov-ref-type => lpt arr obj

                  ref      v-ref        vu-offset mov-ref
      "obj-name" grp id typ "loc-name""mov-name" type  range en
------|---------|---|--|---|---------|----------|-----|-----|-)

 -vu- LCLEVIS_BH  2  14 lpt  UNIT_2    GR-PIC    lpt   561   1

 -vu- LCLEVIS_BH  2  14 lpt  UNIT_1    GR-PIC    lpt   561   1

%%
  FL205-B  8   BRACK-B     11   HBLOCK_BH    15   BLOCK_BH 18
  FL207-B  9   IGES-B      12   LINKHEAD_BH  16   DOG_BH   19
  FL209-B 10   VALVM-B     13   LINKBAR1_BH  17
               LCLEVIS_BH  14   LINKBAR2_BH  17
```

Figure VII.15. A View Location Point/Movetable Form

7.7. Approach/Depart Trajectory Form

Approach/depart trajectory form entries specify how an object is to be approached or departed for a particular grip at a particular location. Entries in this type of form may also be used for approach/depart specification for locations or arrays, but it is more common to use them with objects. Figure VII.16 presents an approach/depart form for the locking clevis blank. It is seen that the form is quite similar to that used for intermediate trajectories. Entries for the approach/depart form, however, use the compiling words -appr and -dept, for approach and depart trajectories, respectively, instead of -traj. Approach and depart trajectories for objects must specify the grip of the object that the trajectory is to be used for. The location where the trajectory applies is also specified in the first line of an entry. Note that the wild card symbol may again be used to indicate the trajectory to apply at all unspecified locations. The second two entries in Figure VII.16 show how to specify the approach to the view location for an object.

The path point lines (with the compiling word -ppt) contain the same parameters as those for defining intermediate trajectories. However, for approach and depart trajectories, the path points to move through are usually specified as a sequence of movetable applied to the reference location, rather than as poses. Also, all approach trajectories must end in a stop point line which specifies the parameters to use during the final motion to the goal point. The last line in an approach trajectory entry must therefore end with the stop indicator, rather than a trajectory point.

BLOCK# 465

```
( -appr/-depart  type  "name" {ref-grip#  loc-type  "loc-name"}
    -ppt | fac | max | acc |  dist |  type |   "name"
---------|-----|-----|-----|-------|-------|------------------)
  -dept   obj LCLEVIS_BH 2 $$$
    -ppt   05    40    10    300      mtb    A/D-8

  -appr   obj LCLEVIS_BH 2 $$$
    -ppt   10    10    10    200      mtb    A/D-5
    -ppt   05    06    40    050      stop

  -appr    vu LCLEVIS_BH  2 lpt UNIT_1
    -ppt   04    40    10    200      mtb    D/A+10rwz
    -ppt   04    40    10    050      stop
  -appr    vu LCLEVIS_BH  2 lpt UNIT_2
    -ppt   04    40    10    050      stop
```

Figure VII.16. An Approach/Depart Trajectory Form

VIII. ENTERING DATA FOR A NEW PART: EXAMPLE

This chapter will illustrate the steps required to program all the necessary data for a new part. This includes preliminary activities such as determining the machining sequence, fixturing details, part blank dimensions and the like which must be performed before the command sequence, poses, trajectories, movetables and object data may be programmed for the part. The part which will serve as an example will be the linkbar, shown in Figure VIII.1. This part makes use of the rectangular parts gripper, vision system, and programmable vise, and is a typical prismatic part that requires only one fixturing.

The usual technique for creating new data forms is to copy a current form of the desired type into an available block, and then edit the data within the form. For objects, the entire 24 block area of forms is copied from a current object, and then the form entries are modified appropriately. All the current system data forms are loaded every time the RCS-RUN boot procedure is executed. Therefore, to make sure data for a new part gets loaded, simply modify the load blocks as necessary.

In creating data which describe robot motions and locations, an attempt should always be made to avoid configurations of the robot where the gripper is close to vertical. These configurations cause problems with the inverse kinematic calculations performed by the DEPC software.

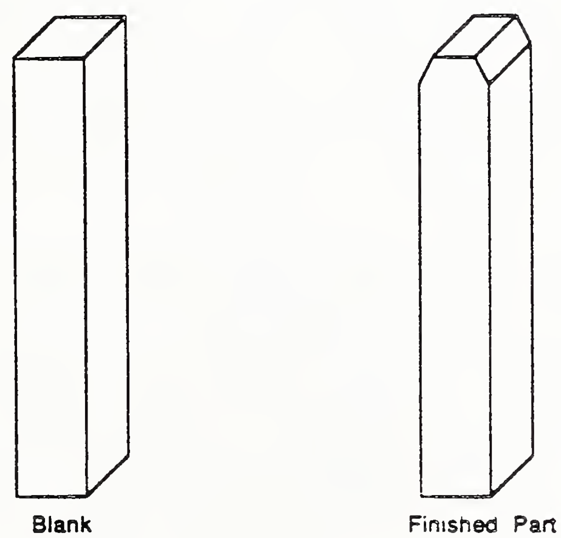


Figure VIII.1. Linkbar Blank and Finished Part.

1. PREREQUISITES

A number of prerequisite details must be specified before data for a new part may be entered. The first of these details is the shape of the part blank. The shape of the part blank will determine the configuration of the tray which will be used to present the blank, whether or not vision will be used to acquire the blank from the tray, which gripper will be used to pick up the blank and how the blank will be held for initial machining (which fixture and where in the fixture). The shape of the part blank is usually determined through an iterative process involving the other workstation personnel. The part blank for the linkbar is a rectangular steel bar with the dimensions 5/8 in. x 5/8 in. x 4 in.

The linkbar blank, a typical prismatic part, will be presented in a standard rectangular parts tray, which is to say that the 21 in. square tray is divided into 4 equal imaginary sectors. Parts in such trays may be placed randomly within a sector, so vision will be required to pick up the part blank from the tray. The rectangular parts gripper will be used for all handling of the part. The part blank will be presented in the tray resting on any of the four long faces. The programmable vise will be used to fixture the part blank. Figure VIII.2 shows the programmable vise jaw profile and where the linkbar is be located during machining.

A general scenario for producing the part may therefore be described. A linkbar blank will be presented to the robot in a sector of a tray at one of the MBD units. The HWSC will command the robot to move this blank from the tray to the programmable vise, which will require that the robot locate the part in the tray using vision. The vise jaws will close on the part blank, but, because of the imprecise location of the blank in the robot gripper, it will not be properly positioned in the vise.

To properly locate the blank for machining, a "hold and seat" procedure is used. That is, the robot will be commanded to move to a "hold" position in front of and close to, but not touching, the part. The vise jaws will then open, allowing the blank to drop and locate accurately on the vise jaw ledge. The robot gripper in the hold position keeps the part from falling out the front of the vise. Next, the robot moves to the "seat" position, which seats the part against the back of the vise. The part is then accurately positioned for the vise jaws to close on it for machining.

After the blank is properly located in the vise and the jaws have closed, the robot will be commanded to move to a safe place and

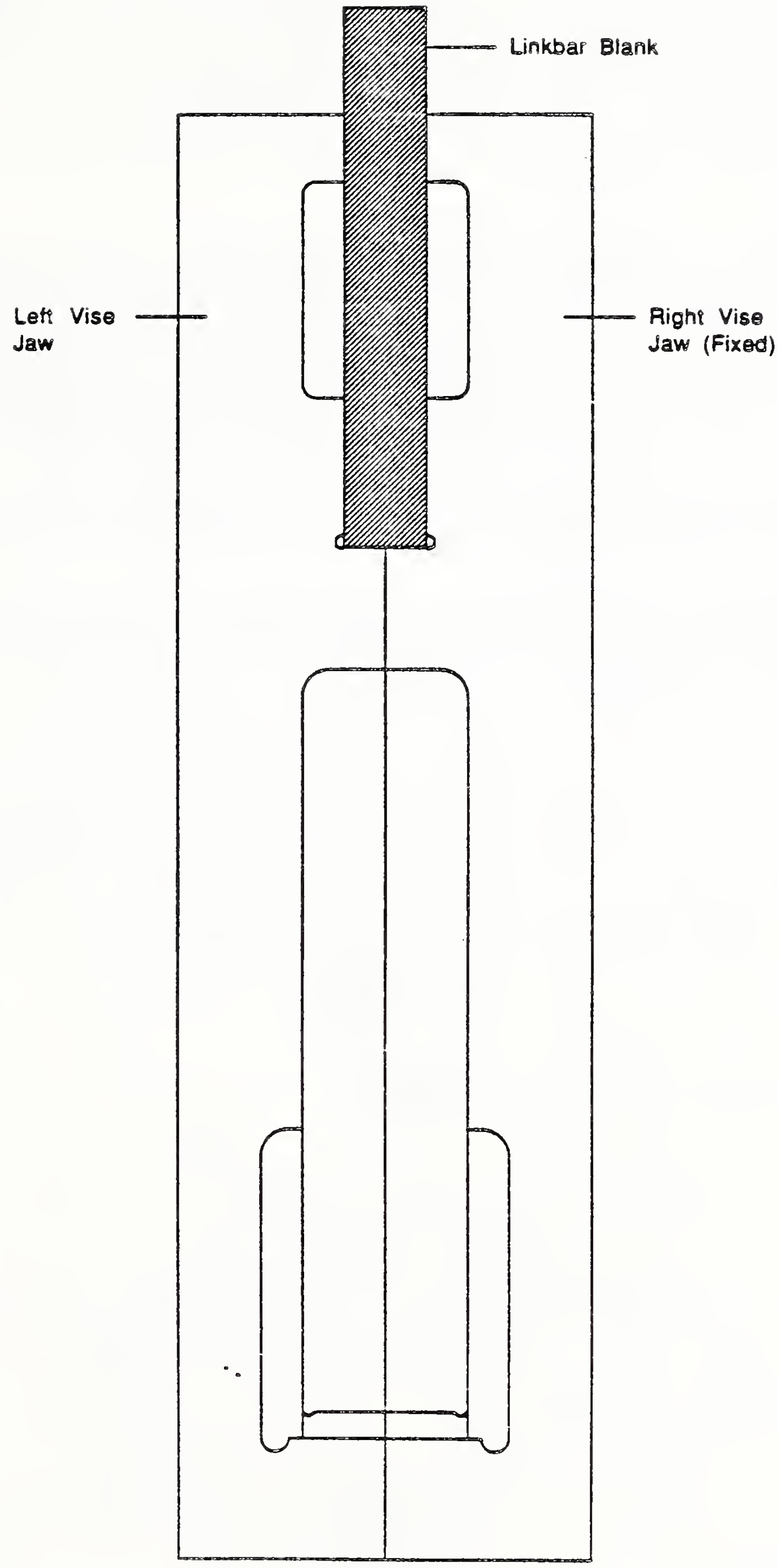


Figure VIII.2. Location of Linkbar Blank in Programmable Vise

wait for machining to take place. When the part is finished, the robot will be commanded to move to the vise and grasp the part. After the vise has released the finished part, the HWSC will command the robot to transfer it to an MBD tray, and end at a safe location to wait for the next command.

2. COMMAND SEQUENCE

The command sequence can be determined from the scenario described above. To specify the commands completely, names are needed for the part blank, the finished part, "virtual" "hold" and "seat" parts, and all locations used. The names of the linkbar blank and finished part, supplied by the administrative systems of the AMRF, are LINKBAR1_BH and LINKBAR1, respectively. The names for the virtual parts used for the hold and seat sequence are derived from these; H-LINKBAR1_BH for the hold, and S-LINKBAR1_BH for the seat.

Figure VIII.3 shows the names of commonly used locations which have already been entered in RCS-II/T3. The locations needed for

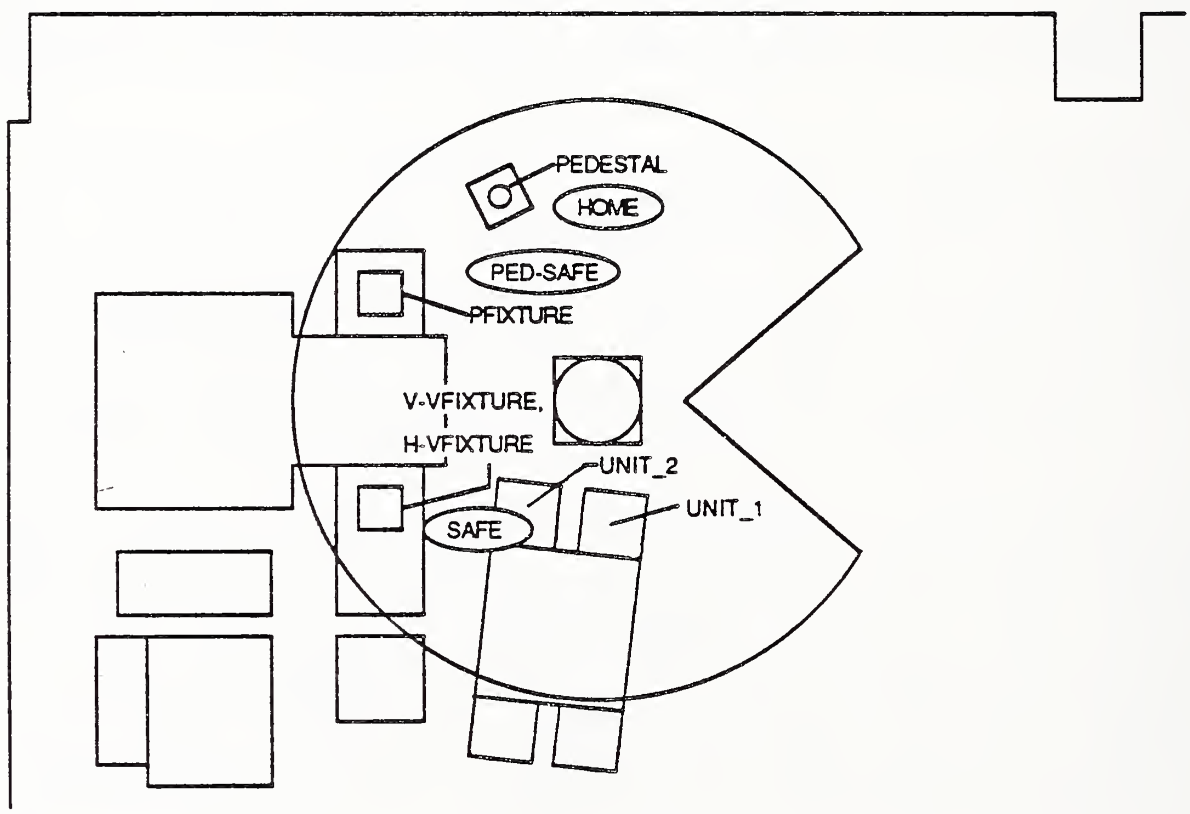


Figure VIII.3. Existing RCS-II/T3 Locations

the linkbar are UNIT_1, UNIT_2, PFIXTURE, HOME, and SAFE. HOME and SAFE are both locations where it is safe to wait for the next HWSC command. The robot is safe in either of these two locations in the event of a power failure or Emergency Stop; the robot should not crush anything while drifting down from either location. It is usually desirable to move the robot to the closer of these two positions while waiting.

The grip numbers to be used for moving the part from one location to the next must also be identified. Following the convention for parts to be located by vision (see Chapter VII, section 7.6), the linkbar will always be handled using grip number +2. Parts with more complicated handling sequences, of course, may need to have numerous grip numbers defined.

The command sequence which will be sent from the HWSC for a linkbar, then, may be set forth. The command sequence for a part is edited into an available block or number of blocks in the area 870-899 at offset 16000. This listing of commands is provided to HWS personnel for HWSC programming, and is also used to execute the commands from a terminal in stand-alone mode, to test them out. The command sequence for the linkbar will be entered in block 893. Figure VIII.4 shows this block after the commands for the linkbar have been entered. The word == enters the command into the Task command buffer when the line is loaded. It is seen (with, perhaps, some reference to Chapter V., section 4) that this sequence will perform the operations described in the scenario above. Double hyphens are used for fields with null values (blanks).

BLOCK# 893

```
\ --------------------Link bar1--------------------------------

 \ Set vise partially closed
== MOVE  LINKBAR1_BH   +2 UNIT_1    4 +2   PFIXTURE 0 -- 0
 \ Close programmable vise
== POSITION  H-LINKBAR1_BH +1 PFIXTURE  0 +0 --         0 -- 0
 \ Open vise to let part drop
== POSITION  S-LINKBAR1_BH +1 PFIXTURE  0 +0 --         0 -- 0
 \ Close vise
== MOVE      --          +0 --        0 +0 HOME       0 -- 0
 \ Do machining
== ACQUIRE  LINKBAR1  +2 PFIXTURE  0 +0 --         0 -- 0
 \ Open vise
== TRANSFER LINKBAR1  +2 PFIXTURE  0 +2 UNIT_1    1 SAFE 0

\ ----------------Done with link bar -----------------------
```

Figure VIII.4. Linkbar Task Command Sequence

3. POSES

As mentioned previously, a number of location points are required to make the linkbar. Poses are required to define these locations. Poses will also be needed to define intermediate trajectory points between these locations. The poses necessary for making the linkbar have already been defined (they are the same as those used for making other prismatic parts). However, if they were not, each would be determined by moving the robot to the desired position with the Joystick Box, pressing the record button, and copying the coordinates displayed into a pose form.

The exact position of the poses defined for the programmable vise and for the MBD units will be important in the discussion of other data. The position of the robot fingers for the programmable vise pose is shown in Figure VIII.5. The pose is taught with the rectangular parts gripper, with the fingers closed and with the gripper axis perpendicular to the front of the vise. The center of the robot fingers is located at the upper left-hand corner of the right (fixed) vise jaw, and the most distal surface of the fingers is against the front surface of the jaw. This pose is named PFIX-P, in block 632, with the movetable named PFIX-COR available for small adjustments to the pose.

The poses for the MBD units is also taught with the rectangular parts gripper with the fingers closed. These poses are taught with the gripper axis perpendicular to the bottom of the tray. The position of the tool frame relative to the tray for these poses is shown in Figure VIII.6. The center of the fingers is positioned over the upper left-hand corner of the tray (from the robot's point of view), with the most distal finger surfaces just touching the top edge of the tray sides. These poses are named UT-1-P and UT-2-P, entered in block 612.

The pose names used for the locations HOME and SAFE are, respectively, HOME-POS and SAFE-P. Several poses used for the intermediate trajectories needed for the linkbar (see section 7) are also required. These are key positions, or via points, taught to move the robot safely through congested areas such as in front of the machine tool. These trajectory poses are not listed here, but may be found in blocks 620-629.

4. MOVETABLES

A large number of movetables must be entered for even a simple part, such as the linkbar. These include offsets needed for any poses, location points, arrays, and different grips, as well as movetables used to define trajectory points for approach and

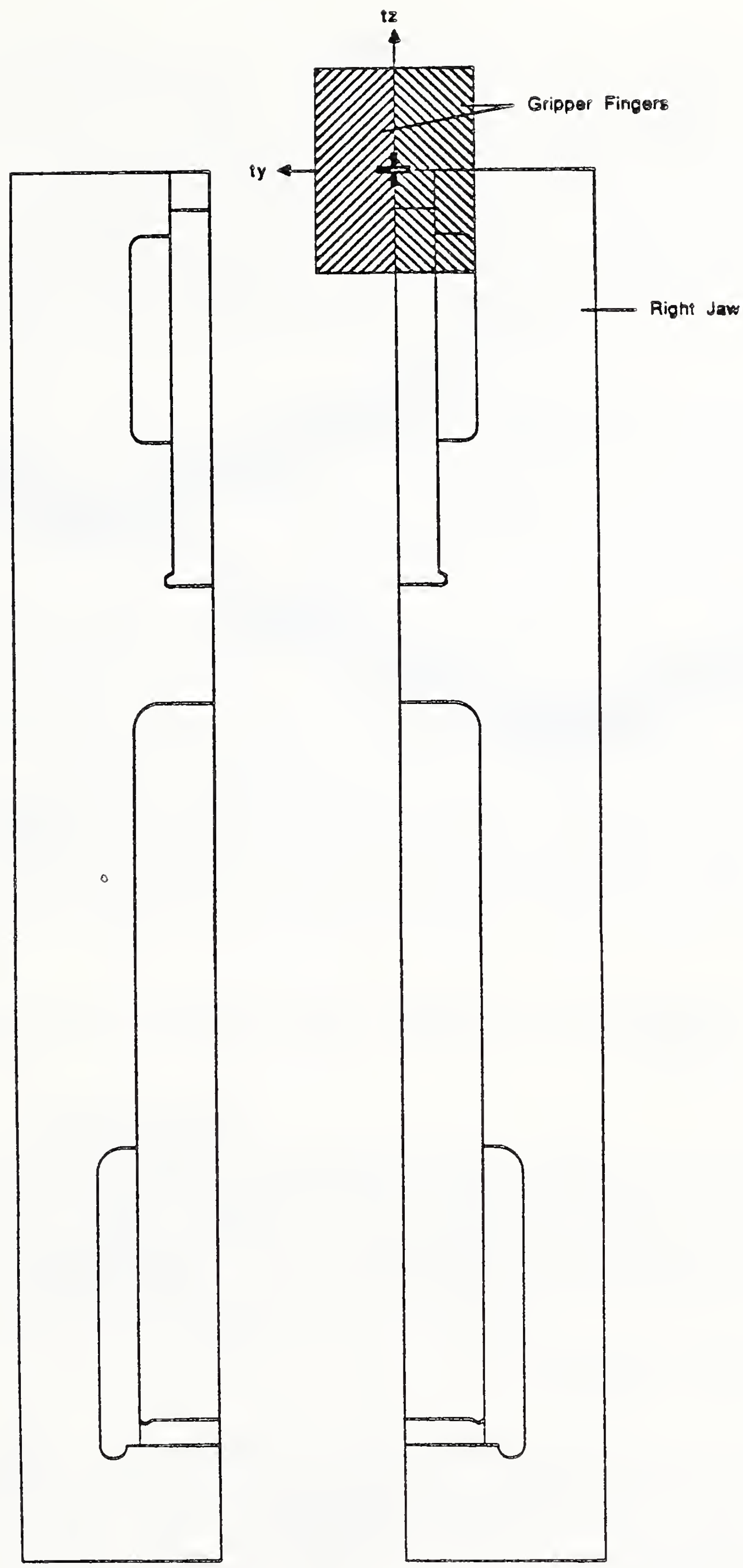


Figure VIII.5. Location of Robot Fingers for Programmable Vise Pose

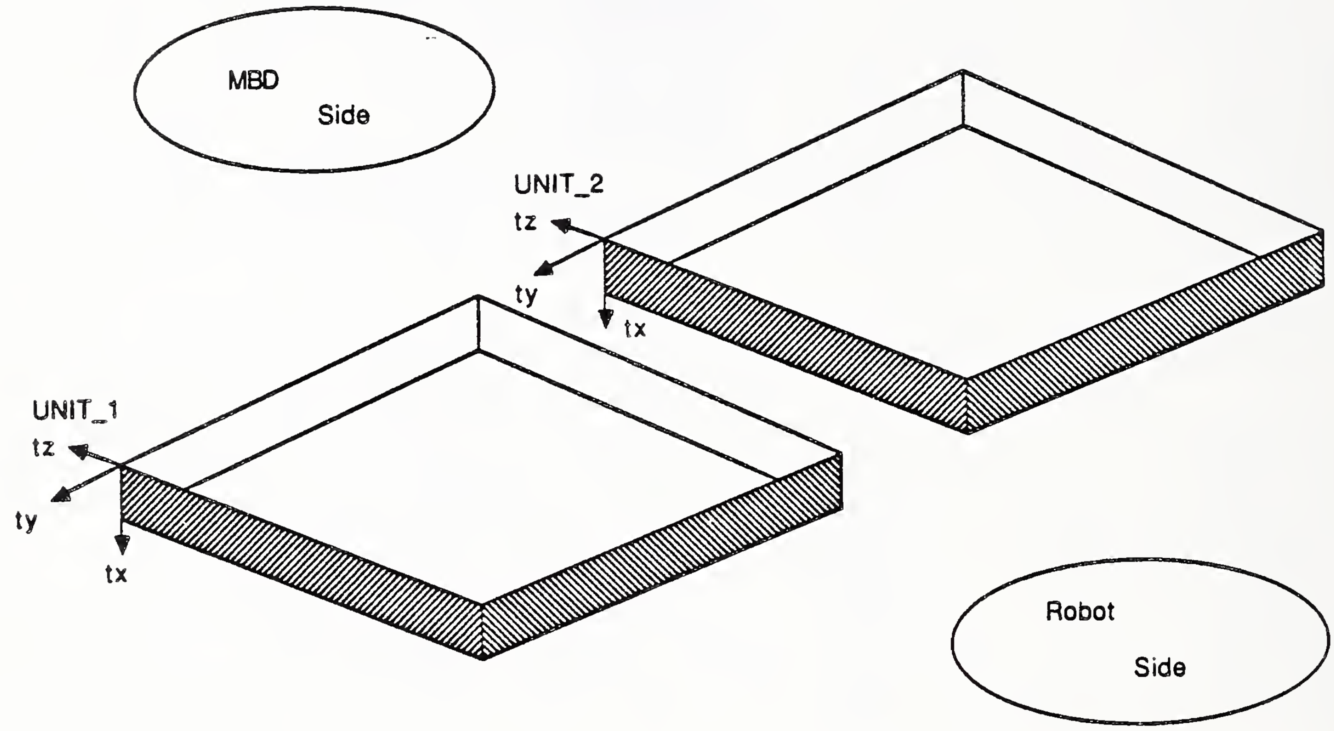


Figure VIII.6. Location of Robot Tool Frame for Unit_1 and Unit_2 Poses

depart trajectories. The movetables for grips and approach/ depart trajectories are of primary interest in entering the data for an object, and the movetables to serve these purposes for the linkbar will be discussed in section 9. The movetables for the tray array used with the linkbar are described in section 6.

5. LOCATION POINTS

The location points needed for the linkbar have already been described. They are entered using the location point form shown in Figure VII.4 in the previous chapter. To add a new location point, simply create an entry in such a form, naming the base pose and the movetable.

6. ARRAYS

The array named SECTORS is the array used for most rectangular parts, including the linkbar. This array is already defined, but the procedure used to create it will be described to show the required steps. The definition of this array requires the completion of two types of array forms and the specification of two movetables. First, an array dimension form is completed (Figure VIII.7), which creates an array named SECTORS. The array is to be two-dimensional, with two elements in each dimension, resulting in a 4-sector division of the tray.

The movetables which define the offset between elements in each dimension must also be named in this form. For SECTORS, we desire four equal, square sectors in a 21 in. square tray. The location of the center of the first sector of the array will be defined such that the robot tool frame is positioned as shown in Figure VIII.8, with the frame resting on the tray surface. The movetable which defines the spacing between the array columns should therefore specify an offset of -10.50 in. in the Y direction in tool coordinates. We will need to create such a movetable, and we may call it -1050ty. Similarly, an offset of -10.50 in. in the tool Z direction is specified by the movetable -1050tz for the spacing between rows in the array. These movetables are entered using the movetable form entries shown in Figure VIII.9. Note that any type of movetable may be specified for array offsets, although single-direction translations are most common.

BLOCK# 702

```
( ARRAY-dim FORM
        |              | # | # | # |     1st     |     2nd     | #
        |"array-name"| dim| 1st| 2nd| "movtab"  |  "movtab"  |pres
        |------------|----|----|----|-----------|-----------|----)

 -arr-  SECTORS        2    2    2    -1050ty     -1050tz     0

 -arr-  TL-SECTORS     1    3          -800ty                 0

 -arr-  TRNG-SCTRS     2    2    4     -840tz       500ty     0
```

Figure VIII.7. Array Dimension Form for SECTORS

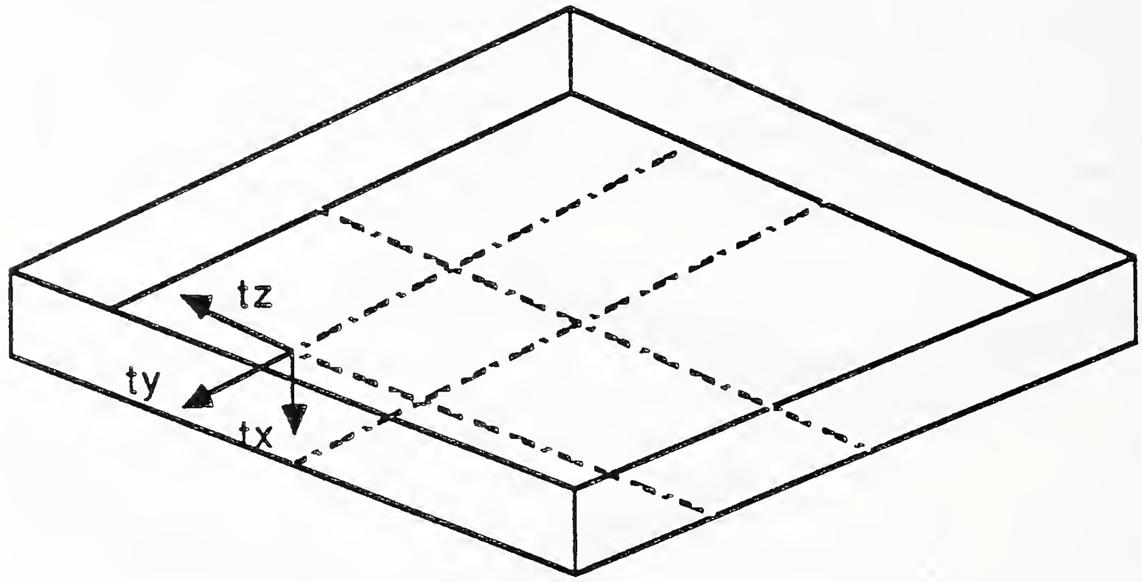


Figure VIII.8. Location of Robot Tool Frame for First Sector of SECTORS

BLOCK# 435

```
ERASE-MOVTAB ( ------------------------------------------------
!    ! r,t    !    x     !    y     !    z     !       !
! ln ! to,wo  !   pit    !   yaw    !          !  ---  !
! #  ! ro     !   roll   !          !          ! -mtb- !  "name"
!----!--------!----------!----------!----------!-------!-----------)

   1      t          0      -1050          0    -mtb-   -1050ty
   1      t          0          0      -1050    -mtb-   -1050tz
   1      t          0       -900          0    -mtb-    -900ty
   1      t          0          0       -900    -mtb-    -900tz
   1      t        -15          0          0    -mtb-     -15tx
   1      t       -400          0          0    -mtb-    -400tx
   1      t       -500          0          0    -mtb-    -500tx
   1      t          0         20          0    -mtb-    +20ty
   1      t          0        -20          0    -mtb-    -20ty
```

Figure VIII.9. Movetables for SECTORS Offsets

An array location form is also needed for SECTORS. This form, shown in Figure VIII.10, names the movetable which gives the location of the first sector when applied to the location point where the array is used. The SECTORS array will be used at UNIT_1 and UNIT_2. As mentioned previously, these location points are defined at the upper left-hand corner of a tray at each MBD unit. For the SECTORS array, the array location movetable must translate this point to the center of the first sector. The movetable which performs these translations will be called SCT-MTB. This movetable is given in Figure VIII.11. The translations performed are: 2.55 in. in the tool X direction to bring the point from the top of the edge of the tray down to the tray surface, and -5.25 in. in each of the tool Y and Z directions to move the point to the center of the first sector (see Figure VIII.8). The SECTORS array is completely defined after the array and movetable form entries have been made.

7. OWNERS

For the linkbar, three owners are needed. First, UNIT_1 and UNIT_2 are grouped in an owner called UNIT so individual intermediate trajectories aren't required to be defined between each unit and the other locations. Second, UNIT_1 and UNIT_2 are both defined as owners, to allow the use of different arrays at the MBD as discussed in the previous chapter. These owners have already been defined. If it is desired to add an array for a new type of tray configuration, the name of the array is simply added to the member list of the owners, UNIT_1 and UNIT_2.

BLOCK# 711

```
( -ar-        "loc-name"      "pose-name"     "movtab-name"
 -----------|--------------|-------------|---------------)

 \ SECTORS is array to be accessed at UNIT_X + SECT-MTB

  -ar-        SECTORS            ...          SCT-MTB

  -ar-        TL-SECTORS         ...          TL-SCT-MTB

  -ar-        TRNG-SCTRS         ...          TRNG-SMTB
```

Figure VIII.10. Array Location Form for SECTORS

BLOCK# 431

```
ERASE-MOVTAB ( ---------------------------------------------------
!    ! r,t    !    x     !    y     !    z     !       !
! ln ! to,wo  !   pit    !   yaw    !          !  ---  !
! #  ! ro     !   roll   !          !          ! -mtb- !  "name"
!----!--------!----------!----------!----------!-------!-----------)
  1    t        255       -525       -525      -mtb-   SCT-MTB
\  offset from tray corner to first sector of 4-sector tray
  1    t          0       -250      -1060       ---
  2    ro        70          0          0       ---
  3    to      -600          0          0       ---
  4    t       -265          0          0      -mtb-   TL-SCT-MTB
\  offset from tray  corner to first sector of tool tray
  1    t        117       -405       -300       ---
  2    ro     -9000          0          0       ---
  3    to         0          0          0      -mtb-   TRNG-SMTB
\  offset from corner to first of 8 sectors of turned-part tray
```

Figure VIII.11. Movetable Form Entry for SCT-MTB

8. INTERMEDIATE TRAJECTORIES

To make the linkbar, intermediate trajectories are needed for the following source and destination location pairs:

HOME -> UNIT
SAFE -> UNIT
UNIT -> PFIXTURE
PFIXTURE -> HOME
HOME -> PFIXTURE
PFIXTURE -> UNIT
UNIT -> SAFE

These trajectories have all been defined in blocks 850-869, using appropriate intermediate trajectory poses taught in the manner described previously. It is important that, to the extent possible, intermediate trajectory points cause the robot to avoid configurations where the robot gripper is vertical.

9. OBJECT DATA

The data described in the previous sections is generic data; it may be useful for other parts. This section discusses the data entered expressly for the linkbar. This data must be entered in a spare 24-block area at offset 20000. The linkbar data will use blocks 251-275. Blocks 0 and 200 are the directory/load blocks which need to be edited to reflect the addition of the linkbar

data. Block 251 will serve as the directory/load block for the 24-block area. This block is shown in Figure VIII.12. Parentheses are used as an alternate commenting symbol; anything enclosed within parentheses is ignored when the block is loaded.

BLOCK# 251

```
  ( Data-forms : Objects :   LINKBAR1
-----------------------------------------------------------------)
   251 ref-blk

( 252 load  (           )    253 load  ( grips     )
  254 load  ( hold/seat)     255 load  ( obj-grip )
  256 load  ( o-loc      )  ( 257 load  (           )
( 258 load  (           )   ( 259 load  (           )
  260 load  ( grips      )    261 load  ( hold/seat)
  262 load  ( vu         )  ( 263 load  (           )
( 264 load  ( a/d 1 buf)     265 load  ( a/d $$$   )
  266 load  ( a/d 1 fix)     267 load  ( a/d 1 fix)
( 268 load  (           )    269 load  ( a/d $$$   )
  270 load  ( a/d ut_2 )     271 load  ( a/d hold )
  272 load  ( a/d seat )   ( 273 load  (           )
( 274 load  (           )   ( 275 load  (           )
```

Figure VIII.12. Directory/Load Block for Linkbar

9.1. Object Form

First, object form entries for the two objects needed for the linkbar must be created. These are shown in Figure VIII.13. The gripper approach size specified for both the linkbar blank and the finished part (with grip number +2) is 3.00 in. This is the opening the part gripper will use when approaching the object to pick it up. The grasp size specified for the linkbar objects is 0.60 in. This is slightly smaller than the actual part size of 0.625 in. The gripper opening to be used when the gripper releases and departs from the linkbar is 3.50 in. The approach and depart sizes are based on clearance, gripping time, and part position uncertainty considerations, and are only relevant for the rectangular parts gripper. The grasp force to be used is set at 30 lb. This value is determined based on the weight of the part, and is empirically adjusted. Values of 30-60 lb. are typical. The grasp force is also only relevant for parts handled by the rectangular parts gripper.

BLOCK# 253

```
( OBJECT FORM
                              ref   appr  grasp  grasp  depart
              "obj-name"     grip#  size  size   force  size
         |----------------|-----|-----||-----||-----||------)

 -obj-   LINKBAR1_BH         2    0300   0060    30    0350

 -obj-   LINKBAR1            2    0300   0060    30    0350
```

Figure VIII.13. Object Form Entries for Linkbar

Object form entries must also be made for the virtual hold and seat objects. These are presented in Figure VIII.14. The control system cannot distinguish between real and virtual objects. For each of these objects grip number 1 will be the only grip used. The approach, grasp, and depart sizes will all be 0.10 in.; i.e. the gripper will be closed. The closing force for the virtual objects is specified as 40 lb.

BLOCK# 254

```
( OBJECT FORM
                              ref   appr  grasp  grasp  depart
              "obj-name"     grip#  size  size   force  size
         |----------------|-----|-----||-----||-----||------)

 -obj-   H-LINKBAR1_BH       1     10     10     40      10

 -obj-   S-LINKBAR1_BH       1     10     10     40      10
```

Figure VIII.14. Object Form Entries For Linkbar Virtual Objects

9.2. Object Grips File

Next, provisions must be made to add two entries to the OBJ-GRIPS file; one for each of the real objects. This is done as shown in Figure VIII.15. The rectangular parts gripper (endeff id = 1) will be used for both linkbar objects, and grip number 2 will be used at the trays.

BLOCK# 255

```
FDEF
:S
          "object"    <=  LINKBAR1_BH
          endeff-id#  <=  1
          tray-grip#  <=  2
:R        >FILE    OBJ-GRIPS


:S        "object"    <=  LINKBAR1
          endeff-id#  <=  1
          tray-grip#  <=  2
:R        >FILE    OBJ-GRIPS
```

Figure VIII.15. OBJ-GRIPS File Entries for Linkbar

9.3. Object Location Form

The object location form entries shown in Figure VIII.16 are used to identify the linkbar objects with the array to be used at the MBD units. Note that a different type of array could be used for the part blank and the finished part, if desired.

BLOCK# 256

```
( OBJ-Loc  FORM      assigned-type => lpt arr
-----------------


                                   assigned        assigned
                 "object-name"     loc-type        "loc-name"
               |----------------|-------------|----------------)

 -o-loc-        LINKBAR1_BH          arr          SECTORS

 -o-loc-        LINKBAR1             arr          SECTORS
```

Figure VIII.16. Object Location Form Entries for Linkbar

9.4. Pedestal Parameter File

The linkbar does not require use of the Active Pedestal, so no entry need be made to the PED-PARA file. The parts named dog, hinge block, and locking clevis provide example command sequences, PED-PARA record entries, and other data for using the Active Pedestal.

9.5. Grip Location point/Movetable Form

A different grip location point/movetable form entry must be created to identify the movetable to be used for every grip of every object at every location. Figure VIII.17 shows the grip location point/movetable form used for the linkbar blank and finished part. For the linkbar blank, two different grip movetables are used. LK-GRP-1 is the default movetable to use when the blank is gripped with grip number 2. This movetable will be used at UNIT_1 and UNIT_2. This movetable designates the offset to use between the location specified by the vision system as the object centroid, and the location where the part is to be gripped.

Since most of the linkbar blank will be down in the vise when the robot moves it there (Figure VIII.2), the blank should be gripped up near the top. There is about 0.75 in. of the part available for grasping when the part is in the vise. If the blank is gripped 2.10 in. away from the centroid in the tool Z direction, as shown in Figure VIII.18, there will be at least 0.10 in. clearance between

BLOCK# 260

```
(  GRIP-LOCPT-movtab FORM     ref-type => lpt arr '$$$
                           ref  ref      ref          grip
            "obj-name"    grip# type  "loc-name"   "mov-name"
 ------|----------------|---|------|------------|------------)

 -grip-  LINKBAR1_BH       2    $$$                LK-GRP-1

 -grip-  LINKBAR1_BH       2    lpt    PFIXTURE    LK-GRP-FX1


 -grip-  LINKBAR1          2    $$$                LK-GRP-1
 -grip-  LINKBAR1          2    lpt    UNIT_2      LK-GRP-1

 -grip-  LINKBAR1          2    lpt    PFIXTURE    LK-GRP-FX2
```

Figure VIII.17. Grip Location Point/Movetable Form for Linkbar

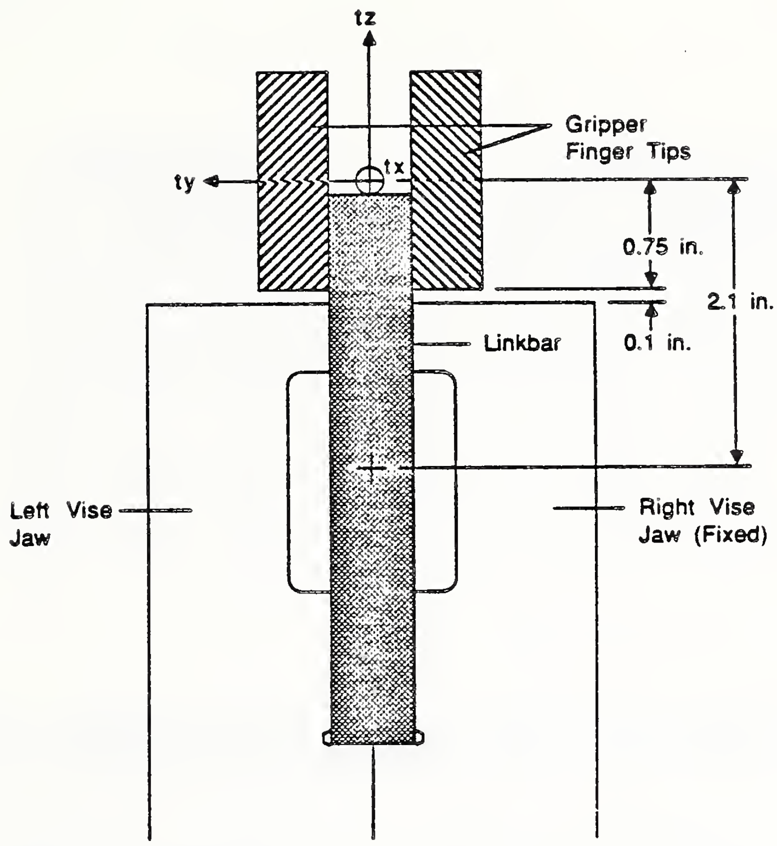


Figure VIII.18. Grip of Linkbar at the Programmable Vise

the 1.50 in. wide gripper fingers and the top of the vise. There are two other considerations for picking up the part blank from the tray. The first is that the gripper finger tips should be raised somewhat above the tray surface, say 0.20 in. The second is that the gripper should be tilted back a bit--3 or 4 degrees will do--about the tool pitch (Y) axis to avoid a vertical configuration. The movetable which incorporates these transformations is shown in Figure VIII.19, along with the other grip movetables used for the linkbar.

Next, a grip movetable must be specified to use at the programmable vise which gives the offset from the position of the gripper holding the blank to the PFIXTURE location defined for the vise. This movetable will be called LK-GRP-FX1 (Figure VIII.17). This offset should position the gripper such that the linkbar blank is a bit above the horizontal vise ledge and centered between the vertical ledges of the vise jaws. Also, the part must be placed at the correct depth in the vise. These requirements will be met if the location taught for PFIXTURE is translated by 0.15 in. in the tool X direction (in toward the vise), 0.19 in. in the tool Y direction (to the left), and 1.00 in. in the tool Z direction (up). Since the part was picked up

BLOCK# 457

```
ERASE-MOVTAB ( ---------------------------------------------------
! ln#! rw,t  !    x     !    y     !    z     ! -mtb- !  "name"
!----!-------!----------!----------!----------!-------!-----------)
  1     t       -20         0        210       ---
  2     to     -300         0          0      -mtb-    LK-GRP-1
  1     t        15        15        125       ---
  2     to     -300         0          0      -mtb-    LK-GRP-FX1
  1     t        05        15        100       ---
  2     to     -300         0          0      -mtb-    LK-GRP-FX2

  1     t       -35         5        -90       ---
  2     ro     9000         0          0      -mtb-    LK-H-GRP

  1     t       -20         5        -90       ---
  2     ro     9000         0          0      -mtb-    LK-S-GRP
```

Figure VIII.19. Grip Movetables for Linkbar

with an angle of 3 degrees, this must also be compensated for in the movetable used at PFIXTURE. The entire LK-GRP-FX1 movetable, then, is as shown in Figure VIII.19.

Similarly, the movetable for gripping the finished linkbar at the programmable vise is LK-GRP-FX2, and the one used at unspecified locations is the same as for the blank, LK-GRP-1 (Figure VIII.17). These movetables are also shown in Figure VIII.19.

The grip movetable/location point form for the virtual objects is shown in Figure VIII.20. For the hold, the movetable LK-H-GRP gives the offset from PFIXTURE to the hold position. LK-S-GRP gives the offset from PFIXTURE to the seat position. For the hold and seat positions, the robot gripper will be rotated 90 degrees so the gripper fingers will contact across the part. There is no need to pitch the gripper, since it will not be holding a part grasped at an angle or be near vertical. The LK-H-GRP shown in Figure VIII.19 provides first the translations to move PFIXTURE to a position down in front of the part, and then the 90 degree rotation. As with all movetables, these successive transformations are applied to produce a single location which the robot goes to (the lines in a movetable are not and cannot be moved to independently). The seat grip movetable is identical to that for the hold, except that the seat position contacts the part and pushes it in against the back of the vise. This is evidenced by the 0.15 in. difference in the tool X direction.

BLOCK# 261

```
( GRIP-LOCPT-movtab FORM     ref-type => lpt arr $$$
                      ref   ref       ref         grip
        "obj-name"   grip#  type  "loc-name"   "mov-name"
------|---------------|---|------|-----------|-----------)

 -grip-  H-LINKBAR1_BH   1   lpt    PFIXTURE      LK-H-GRP

 -grip-  S-LINKBAR1_BH   1   lpt    PFIXTURE      LK-S-GRP
```

Figure VIII.20. Grip Location Point/Movetable Form for Linkbar Virtual Objects

9.6. View Location Point/Movetable Form

Two view location point/movetable entries are needed for the linkbar blank, one for each MBD unit. As seen in Figure VIII.21, the two entries are identical except for the location name. The reference grip number for the view is 2 (the part will be picked up with this grip), the vision id number for the linkbar blank is 17, the location type is location point, denoted by lpt. The movetable named GR-PIC is currently used for all parts requiring vision. It defines the movetable which, when applied to the location of a sector, properly positions and orients the camera for picture-taking. The range estimate is measured from the image plane of the camera to the viewed surface of the part. When GR-PIC is applied to a location at the tray surface, the range to the surface is about 605 mm. The estimated range to the top of the part may therefore be estimated by subtracting the height of the part from this value. The enable flag is set to 1, so vision will be used for the part.

9.7. Approach/Depart Trajectory Form

The last information which needs to be entered for the linkbar is the collection of trajectory forms and movetables that defines how the robot will approach and depart the linkbar objects at each location. The reason for designating specific approach and depart trajectories is to avoid obstacles and perform tasks such

BLOCK# 262

```
( VIEW-LOCPT-movtab FORM      v-ref-type => lpt arr
                              mov-ref-type => lpt arr obj

      ref        v-ref          vu-offset mov-ref
    "obj-name"   grp id typ "loc-name" "mov-name" type range en
----|------------|--|--|---|----------|----------|----|----|--)

 -vu- LINKBAR1_BH  2 17 lpt  UNIT_1     GR-PIC     lpt   590 1

 -vu- LINKBAR1_BH  2 17 lpt  UNIT_2     GR-PIC     lpt   590 1

\\
   FL205-B  8   BRACK-B    11   HBLOCK_BH   15   BLOCK_BH 18
   FL207-B  9   IGES-B     12   LINKHEAD_BH 16   DOG_BH   19
   FL209-B 10   VALVM-B    13   LINKBAR1_BH 17
                LCLEVIS_BH 14   LINKBAR2_BH 17
```

Figure VIII.21. View Location Point/Movetable Form for Linkbar

as part insertion into a fixture correctly. There are a number of movetables located in blocks 520-529 at offset 21000 which are used for approach and depart trajectories. It is most convenient if an acceptable trajectory may be created using these previously-defined movetables. This is the case with the linkbar; however, there is room in this area for new movetables to be added when necessary.

First, we shall define the default approach and depart trajectories to be used for the linkbar blank. These will be used at UNIT_1 and UNIT_2. To approach the blank in the tray, we wish the robot to first move fairly rapidly through a position somewhat far above the part, say 10 in., and then, more slowly, to a position closer to the part, maybe 5 in. away. Finally, the robot should move quite slowly to the final destination. These motions are provided by the second entry in the approach/depart trajectory form shown in Figure VIII.22. The movetable APPR-1 is a -10.00 in. translation in the tool X direction, while APPR-2 designates a -5.00 in. translation in the same direction. Approach and depart trajectory movetables are applied after all other movetables (array, vision, grip, etc.) which determine a location. Therefore, when passing through the approach points to pick up a linkbar blank, for example, the robot will be positioned over the part location as modified by vision, and tilted back the 3 degrees specified by the grip movetable.

To depart from the tray with the blank, we will have the robot

```
BLOCK# 265
( -appr/-depart  type  "name" {ref-grip#  loc-type  "loc-name"}
    -ppt | fac | max | acc |  dist |  type |    "name"
---------|-----|-----|-----|-------|-------|------------------)
  -dept   obj LINKBAR1_BH 2 $$$
    -ppt   05    40   ¯10    300      mtb    A/D-8

  -appr   obj LINKBAR1_BH 2 $$$
    -ppt   10    10   ¯10    200      mtb    A/D-5
    -ppt   05    05    40    050      stop

  -appr    vu LINKBAR1_BH  2 lpt UNIT_1
    -ppt   04    40   ¯10    200     ¯mtb    D/A+10rwz
    -ppt   04    40    10    050      stop
  -appr    vu LINKBAR1_BH  2 lpt UNIT_2
    -ppt   04    40   ¯10    050     ¯stop
```

Figure VIII.22. Approach/Depart Trajectory Form for Linkbar

pass through a single point 8.00 in. above the tray. This is shown in Figure VIII.22. Also shown in this figure are trajectories which define how to approach the view location point (type vu) at each MBD unit. For UNIT_1, the view location is approached through a position 10.00 in. above in real world coordinates (i.e. directly above the view point). This extra height is needed to clear any tools which may be in a tray at UNIT_2. For UNIT_2, the view point is moved to directly.

Next, approach and depart trajectories need to be specified for the linkbar blank at the programmable vise (Figure VIII.23). To approach PFIXTURE with the blank, the familiar APPR-1/APPR-2 approach is used, but an additional approach point, 0.25 in. above the final destination, is moved through in the last approach segment. This is done to prevent the bottom of the part blank from catching on the ledge of the vise if the part has been gripped a little too high. To depart from PFIXTURE after releasing the part, the DPART-1 trajectory is used. The same approach and depart trajectories are used for the finished linkbar at PFIXTURE, as shown in Figure VIII.24. Note that the robot picks the finished part up by 0.25 in. before removing it from the vise, to avoid dragging the bottom of the part on the vise ledge.

For placing the finished part back in the tray, two sets of approach/depart trajectories are defined, one for UNIT_1 and one for UNIT_2. As with the approach to the view point discussed

```
BLOCK# 266
(    -ppt | fac | max | acc |  dist |  type |   "name"
---------|-----|-----|-----|-------|-------|------------------)

  -dept    obj LINKBAR1_BH  2  lpt  PFIXTURE
    -ppt   05    10   10    50      mtb    D/A+.25rwz
    -ppt   05    40    5   300      mtb    A/D-8

  -appr    obj LINKBAR1_BH  2  lpt  PFIXTURE
    -ppt   05    30    5   500      mtb    A/D-10
    -ppt   05    10   10   200      mtb    A/D-5
    -ppt   05     5   40     0      mtb    D/A+.25rwz
    -ppt   04    05   40     0      stop
```

Figure VIII.23. Approach/Depart Trajectory Form for Linkbar Blank at Programmable Vise

earlier, the approach and depart used for placing a finished linkbar back in UNIT_1 should ensure that the robot moves high enough over UNIT_2 to avoid any tall objects that might be in a tray there. The movetable A/D-15 (-15.00 in. in the tool X direction) is used in these trajectories to provide this height, as shown in the form in Figure VIII.25. The approach and depart for UNIT_1 also serve as the default trajectories for the finished linkbar, as indicated by $$$. It is important to note that in moving around to UNIT_1, the robot can come very close to

```
BLOCK# 267
(    -ppt | fac | max | acc |  dist |  type |   "name"
---------|-----|-----|-----|-------|-------|------------------)

  -appr    obj LINKBAR1       2  lpt  PFIXTURE
    -ppt   05    40    5   500      mtb    A/D-10
    -ppt   05    20    5   200      mtb    A/D-5
    -ppt   05    10   40     0      mtb    D/A+.25rwz
    -ppt   04    06   40     0      stop

  -dept    obj LINKBAR1       2  lpt  PFIXTURE
    -ppt   05     5   40     0      mtb    D/A+.25rwz
    -ppt   05    15   10   200      mtb    A/D-8
```

Figure VIII.24. Approach/Depart Trajectory Form for Finished Linkbar at Programmable Vise

```
BLOCK# 269
( -appr/-depart  type  "name" {ref-grip}  loc-type  "loc-name"}
         | acc | vel | smoo|  brk- |  ppt   |   path-pt
    -ppt | fac | max | acc |  dist |  type  |   "name"
---------|-----|-----|-----|-------|--------|-----------------)
% appr/dept unit_1

  -dept   obj LINKBAR1        2  $$$
    -ppt   05    40    15    200      mtb    A/D-15

  -appr   obj LINKBAR1        2  $$$
    -ppt   05    60    15    150      mtb    A/D-15
    -ppt   05    30    40    150      mtb    A/D-5
    -ppt   04    10    40    050      stop
```

Figure VIII.25. Default Approach/Depart Trajectory Form for Finished Linkbar

software joint limit on the shoulder axis if a point is too high. This possibility should be considered in determining trajectories around the MBD. At UNIT_2, the robot approach and depart can be lower, as exemplified in the use of APPR-2 alone in the trajectories shown in Figure VIII.26.

Lastly, the approach and depart trajectories must be specified for the hold and seat virtual objects. For the hold object, we

```
BLOCK# 270
( -appr/-depart  type  "name" {ref-grip}  loc-type  "loc-name"}
         | acc | vel | smoo|  brk- |  ppt   |   path-pt
    -ppt | fac | max | acc |  dist |  type  |   "name"
---------|-----|-----|-----|-------|--------|-----------------)

  -dept   obj LINKBAR1        2  1pt  UNIT_2
    -ppt   05    40    15    150      mtb    A/D-5

  -appr   obj LINKBAR1        2  1pt  UNIT_2
    -ppt   05    40    15    150      mtb    A/D-5
    -ppt   04    10    40    050      stop
```

Figure VIII.26. Approach/Depart Trajectory Form for Finished Linkbar at UNIT_2

will want the robot to approach the part from a short distance in front of the vise, so we will use APPR-2. To depart from the hold position, we want to specify that no depart points be moved through, since we will only be moving to the seat position a short distance away. These trajectories are shown in Figure VIII.27. Similarly, for the seat object, no approach points are desired, and a single point a short distance from the part will do for the depart trajectory. The trajectory form for these trajectories may then look like Figure VIII.28.

We have now defined all the information required to make the linkbar, including approach and depart trajectories for the linkbar objects at each location. When the data is loaded, all the information will be available to execute the commands for the part, listed earlier, whether executed from the HWSC or line-by-line from terminal.

BLOCK# 271

```
( -appr/-depart  type  "name" (ref-grip#  loc-type  "loc-name")
         | acc | vel | smoo| brk- |  ppt  |   path-pt
    -ppt | fac | max | acc | dist |  type |   "name"
---------|-----|-----|-----|------|-------|-----------------)

 -dept   obj H-LINKBAR1 BH 1  1pt  PFIXTURE
   -ppt   05    10    10   100     mtb     ...

 -appr   obj H-LINKBAR1 BH 1  1pt  PFIXTURE
   -ppt   05    10    10   100     mtb    A/D-5
   -ppt   04    05    04   050     stop
```

Figure VIII.27. Approach/Depart Trajectory Form for Hold Virtual Object

BLOCK# 272

```
( -appr/-depart  type  "name" (ref-grip#  loc-type  "loc-name")
         | acc | vel | smoo| brk- |  ppt  |   path-pt
    -ppt | fac | max | acc | dist |  type |   "name"
---------|-----|-----|-----|------|-------|-----------------)

 -dept   obj S-LINKBAR1 BH 1  1pt  PFIXTURE
   -ppt   05    40    5    300     mtb    A/D-8

 -appr   obj S-LINKBAR1 BH 1  1pt  PFIXTURE
   -ppt   04    05    40   0       stop
```

Figure VIII.28. Approach/Depart Trajectory Form for Seat Virtual Object

IX. EXTERNAL COMMUNICATIONS

This chapter discusses the protocol used by RCS-II/T3 to communicate with external systems. The external systems communicating with RCS-II/T3 include the Horizontal WorkStation Controller (HWSC), the AMRF Data System, the Active Pedestal, and the NBS vision system.

Communications with most external systems occurs through the AMRF factory network. The reader should be familiar with the basic operations of the network as described in AMRF network documentation. The following discussion will assume that the reader has this basic understanding. This discussion will not delve into extensive network details.

From RCS's standpoint, the network mailboxes are segments of common memory of predetermined sizes, in predetermined locations (or addresses). Table IX.1 gives the common memory addresses for the network mailboxes currently used by the system. Generally, communication with an external system involves two types of mailboxes, a command mailbox going one way and a status mailbox going the other. Thus the table shows two mailboxes for each external interface, (except for vision). The two extra status mailboxes for the pedestal provide information to the Diagnostics process in RCS.

Each mailbox complies with a format used throughout the system. This format is depicted in Figure IX.1.

Table IX.1. Addresses of RCS-II/T3 Network Mailboxes

Mailbox Description	Address (in Hex)	Size (in # bytes)
HWSC to RCS Command	F6000	256
HWSC from RCS Status	F6200	256
Data System from RCS Command	F7000	512
Data System to RCS Status	F7300	256
Pedestal from RCS Command	F5000	256
Pedestal to RCS Status	F5300	256
Pedestal Gripper to RCS Status	F5600	256
Pedestal Table to RCS Status	F5900	256

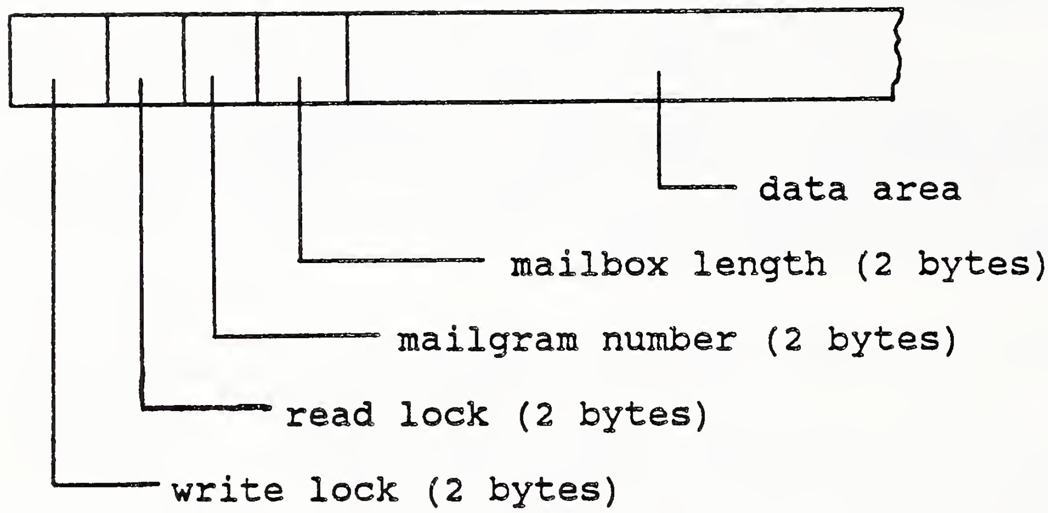


Figure IX.1. Network Mailbox Format

The mailgram number is used to notify the network that a mailbox contains new data and should be sent across the network. Mailboxes which are read, are read every cycle, so that no such indicator is needed for incoming data. Also, since mutual exclusion to a mailbox, i.e. between an RCS process and the network software, is successfully achieved by the bus lock, the read and write lock variables of the mailbox are not used by RCS.

Mailboxes must be initialized by the system during start-up. When mailboxes are initialized, the mailbox length is stuffed along with any preset data. Then the mailgram number and lock variables are stuffed with zeros. Initialization generally occurs during the 28 LOAD of the process board. It is also possible to reinitialize mailboxes by loading the data stuffing blocks used by 28 LOAD. This is often desirable when network communication gets muffed, or when reconnecting a process to the network, e.g. reconnecting Task to the HWSC network commands.

The NBS vision system is the one exception to the rule that communication with external systems occurs through the factory network. The vision system has special facilities for high-speed parallel communications with RCS-II/T3. This results in a

somewhat unique interface for vision which is explained in the final section of this chapter. The first three sections of the chapter cover the specifics of communications with the HWSC, Data System, and Active Pedestal, in that order.

1. RCS AND HWS

Communications with the Horizontal WorkStation Controller (HWSC) from the Task Level of RCS-II/T3 look the same as communications between internal processes of the system. The information content of the HWSC-Task interface is described in V.4.2. of this document. The only functional differences in the interface necessitated by the network involve initialization and data transfer. These two components are described below.

Initialization of the network mailboxes occurs automatically when the system is initialized. The 28 LOAD for Task Level loads block 641 at 16000 OFFSET to do the initialization of the HWSC mailboxes. If the system is brought-up without HWSC connections, and then some time latter HWSC connections are to be made, the operator should first reinitialize the mailboxes by loading block 641 of the Task Level. This assures that HWSC and RCS will synchronize with regard to command numbers. (The process for connecting to HWSC is given in IV.4.1.)

Data transfer to network mailboxes is achieved through the read and write routines of Task Preprocess and Postprocess. These routines use the segment variables defined in blocks 190-199 to do a bus-locked transfer of the data to common memory. The UPDATE-MAILGRAM? routine of Task Postprocess updates the mailgram number on the status mailbox to HWSC when there is new information to be transmitted over the network.

2. RCS AND DATA SYSTEM

Data System communications result in the most complex interface associated with RCS-II/T3. This is because of the complexity of the protocol necessary for data server commands and the fact that the data server uses parsed-string instead of fixed-field commands. The data transfer and initialization techniques used with network mailboxes, however, is the same as with other external interfaces.

The command buffer from Task to the Data System is defined in blocks 182-184 of the Task Level. The command buffer, as shown in Figure IX.2, can be broken into two major blocks, the data server and DML protocol area, and the DML string area.

The protocol area consists of the byte-by-byte values necessary

for the correct command protocol to the Data System. The correct protocol values are specified in the Data System Command and Status Documentation. Most of these values are constant for all Data System commands. Therefore the values are stuffed when the variable owner is initialized, and do not have to be changed when the system is running.

```
BLOCK 182
 TASK DEFINITIONS
 2?? bytes  VAR-0   data-command-var
        iv    DSCindc          HEX    (= 82A0  DECIMAL
        iv    DSClen           HEX    (= 1800  DECIMAL
        iv    IDindc           HEX    (= 0A80  DECIMAL
        iv    command-#)d             (= 00
     8 strv   usedid)d                (= H_ROBOT
        iv    TRNSindc         HEX    (= 8230  DECIMAL
        iv    TRNSlen          HEX    (= 0800  DECIMAL
        bv    TYPEindc         HEX    (= 85    DECIMAL
        bv    TYPElen                 (= 00
        iv    TRNSIDinc        HEX    (= 0450  DECIMAL
     2 strv   trans-id                (= TR
        iv    trans-id#               (= 00
        iv    DMLindc          HEX    (= 8204  DECIMAL
        iv    DMLlen           HEX    (= D000  DECIMAL ( 208)
BLOCK 183
% 106       VAR-0   data-command-var  (cont)
   7  strv   "dbcommand"        (= UPDATE
   18 strv   "report"           (= EQUIP_ITEM_ACTION
   4  strv   "set"              (= SET
   2  strv   "("                (= (
   11 strv   "item-type"        (= ITEM_TYPE,
   11 strv   "item-name"        (= ITEM_NAME,
   8  strv   "device"           (= DEVICE,
   13 strv   "storage-pos"      (= STORAGE_POS,
   7  strv   "sector"           (= SECTOR
   2  strv   ")"                (= )
   2  strv   "="                (= =
   2  strv   "(("               (= ("
   16 strv   "ITEM-TYPE"
   3  strv   ","                (= ','

BLOCK 184
% 102       VAR-0   data-command-var  (cont)
    16 strv   "ITEM-NAME"
    3 strv   ",,"               (= ','
    16 strv   "DEVICE"
    3 strv   ",,,"              (= ','
    16 strv   "STORAGE-POS"
    3 strv   ",,,,"             (= ','
    7 strv   "SECTOR-"
    1 strv   "SECTOR#"
    3 strv   "))"               (= ")
    6 strv   "where"            (= WHERE
    8 strv   "item-nr"          (= ITEM_NR
    2 strv   "=="               (= =
    1 strv   "'"                (= '
    16 strv  "ITEM#"            (= ????????????????
    1 strv   "'"                (= '
```

Figure IX.2. Data System Command Buffer

RCS uses two basic types of commands to the Data System, data server reconfiguration commands and DML commands. Data server reconfiguration commands require only 28 byte lengths, so that DSClen and TRNSlen are modified appropriately. In addition, the correct transaction type must be set in TYPEindc. For DML commands, the entire 240 byte command string is used so that DSClen, TRNSlen, and TYPEindc must be modified to meet the DML requirement. All of this protocol modification is handled automatically by the routines in blocks 411-416.

The DML string area of the command buffer contains the actual data relevant to the DML Transaction. For RCS, this data is an UPDATE via an EQUIP-ITEM-ACTION report. Details of this report and its format can be found in Control-Database Interface Documents. As far as Task is concerned, it only has to stuff in the appropriate values for ITEM-NAME, DEVICE, STORAGE-POS, SECTOR-, SECTOR#, and ITEM#. All of this information should be available from the HWSC command to Task. Thus, the routine REPORT-ITEM, (blocks 417-419,) can stuff these data areas with the information when necessary.

Data System status information, unlike the out-going command data, must be parsed. To achieve this the mailbox is read into a string variable big enough to hold the relevant data. This string buffer is defined in block 107 of the Task Level. A routine in Task Preprocess, namely Parse-das-status, extracts the server-status and Sumry-status bytes, so that the Task Level can determine when an UPDATE has completed.

The Task Level interacts with the Data System network mailboxes in the usual manner. That is, data is transferred to and from the mailboxes by the read and write routines defined in Task Preprocess and Postprocess. The mailboxes are initialized by loading blocks 642 and 643. This is taken care of automatically during system start-up.

3. RCS AND ACTIVE PEDESTAL

The Active Pedestal and RCS-II/T3 systems are built on the same fundamental control structure. Therefore, the communication techniques used from the Pedestal side are very similar to those on the RCS-II/T3 side. From RCS-II/T3's viewpoint, the Pedestal is the mailboxes at the common memory addresses given in Table IX.1. From the Pedestal's point of view, RCS-II/T3 is the corresponding mailboxes given in Table IX.2. (The fact that the addresses for these mailboxes are the same is purely coincidental.)

The mailboxes from the Pedestal provide interfaces to two

Table IX.2. Addresses of Pedestal Network Mailboxes

Mailbox Description	Address (in Hex)	Size (in # bytes)
RCS to Pedestal Command	F5000	256
RCS from Pedestal Status	F5300	256
RCS from Pedestal Gripper Status	F5600	256
RCS from Pedestal Table Status	F5900	256

processes of RCS-II/T3. The overall command and status to the Pedestal allows Subtask to control the Active Pedestal when the robot is regripping parts. The Gripper and Table Status mailboxes are read by the Diagnostics process. This allows the operator to see the states of the Gripper and Table subprocesses of the Pedestal on the graphics display monitor. Both Subtask and Diagnostics use the same techniques for initialization of mailboxes, and for data transfer to and from mailboxes.

Subtask initializes its mailboxes on start-up, by loading block 641 at 17000 OFFSET. Data is transferred by the read and write routines in Subtask Preprocess and Postprocess. These routines use the segment variables defined in block 193 to do a bus-locked transfer of the data between the common memory mailboxes and the local variable buffers. The buffers for Subtask to Pedestal command and status are defined in blocks 182 and 104, respectively.

For Diagnostics, the status buffers are defined in blocks 103-104 at 24000 OFFSET. The segment variables which specify the mailbox addresses are in block 142. These segment variables are used, just as in Subtask, by the read routines called in Diagnostics Preprocess and Postprocess. The read routines transfer the data from the mailbox, where it was delivered by the network, to the status buffers used by Diagnostics to update the graphics. For more details on Diagnostics consult VI.6.

4. RCS AND VISION

The vision system is the only external system with which RCS communicates in a way other than the factory network. Communications with the vision system occur through a high speed 589 parallel interface. As far as the software is concerned, this interface looks like a slightly modified network mailbox. The format for the vision mailbox is depicted in Figure IX.3. The figure includes the addresses (in hex) and the size of each area of the mailbox.

The vision-ready areas are used to synchronize RCS and Vision System Communications. When there is a two ("ready") in vision-ready-input, then the vision response can be read by RCS. When there is a two ("ready") in vision-ready-output, then RCS can write a new request to the vision system. Reading or writing data involves only the bus-locked transfer of data to or from the common memory areas of the vision mailbox. This activity is

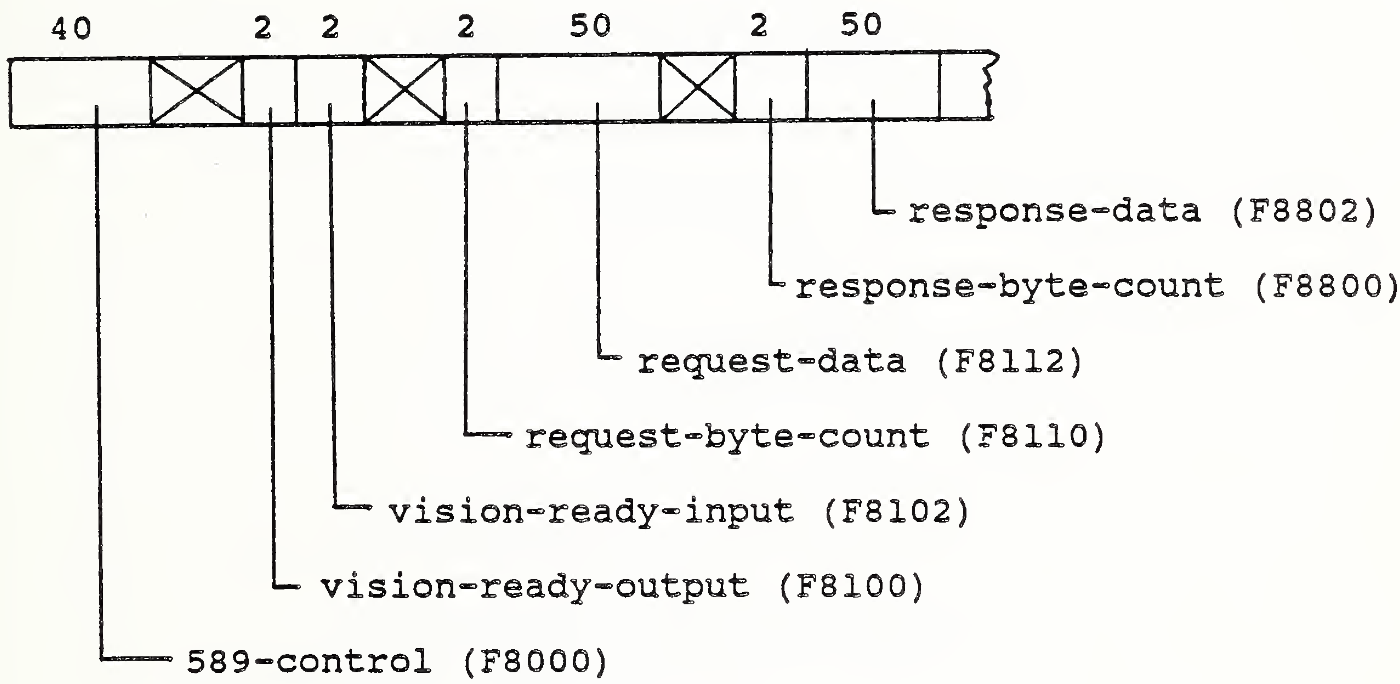


Figure IX.3. Vision System Mailbox Format

carried out by the read and write routines in E-move Preprocess and Postprocess. These routines use the segment variables defined in block 193 at 18000 OFFSET to transfer data to and from the buffers in blocks 103 and 182.

Initialization of the 589 board and the communications mailboxes are taken care of in the 28 LOAD. This is executed when the system starts up and should not have to be re-executed when the system is running. The initialization procedures, blocks 984-986 of E-move, set the byte-counts for the request and response areas. They also set vision-ready-output to "not ready" and vision-ready-input to "ready". The vision system establishes communications at some time subsequent to this initialization.

APPENDIX A

T3 DEPC LINK DOCUMENTATION

"DYNAMIC EXTERNAL PATH CONTROL"

(DEPC)

FUNCTIONAL SPECIFICATION

VERSION 1.1

7 JUL 82

(FOR VERSION 3 CONVERSION ROBOTS)

1.0 OVERVIEW

Dynamic External Path Control (DEPC) is a function which puts the robot into a mode of operation whereby an external computer can dynamically control the path of the robot. Using an RS232 serial communications link, the external computer transmits position and orientation 'world' coordinates to the robot at approximately a 50 Hz rate. By properly spacing the world coordinates, the external computer controls the position, velocity and acceleration of the robot.

Additionally, the external computer can transmit Function Data to the robot, and cause the robot to execute Continue Type functions (DAC or Output).

2.0 OPERATION

2.1 Teaching

(FUNCTION) E OH (ENTER)

E for External
OH for DEPC (that's letter O letter H)

2.2 Teach Mode Execution

None

2.3 Auto Mode Execution

DEPC checks for the presence of the RS232 Communications board. If the board is missing warning DØ is issued.

DEPC turns on its RS232 DTR signal. Then it checks its RS232 CTS signal. If CTS is low or ever goes low while receiving, the external computer is assumed absent, and a warning D1 is issued.

DEPC transmits the following message to the external computer: (FF), (FF), (Ø3), 12 bytes of representing the present world coordinates, and a checksum byte.

DEPC waits to receive one of three types of command:

1) Data Command: (Ø3); 2 function bytes*, 12 bytes of world coordinates, and a checksum byte.

2) Cancel Command: (18); DEPC acknowledges the Cancel Command by transmitting an ACK (Ø6), then terminates the DEPC function and moves to the next programmed point.

3) Re-Start Command: (Ø5); causes DEPC to re-start, ie., transmit the present robot coordinates, and then wait for another command.

DEPC's response to the Cancel Command and Re-start Command are explained above. DEPC responds to a Data Command by first transmitting an ACK (Ø6), and then moving to the received coordinates during the next 19.76 milliseconds.

While the robot is moving to the received coordinates, it can be receiving the next command.

If DEPC can not decipher the command, a NAK (15) will be transmitted to the external computer, and the buffer on the RS232 board will be cleared. Since the buffer should only have one command in it, the external computer can recover, from a NAK by re-transmitting its last command.

The checksum is calculated, by initializing the checksum to zero, and then exclusive or-ing the checksum with all the bytes in the command. Note, Cancel and Re-start Commands do not have a checksum.

If the commanded change in translation or orientation exceeds certain limits, a warning D3 or D4, respectively, is issued. This is a velocity restriction. The limitations prevent the robot from exceeding its capabilities. The translation limit has been set to $\approx$ 14 in/sec. The orientation limit has been set to $\approx$ 1.0 rad/sec.

Limitations also exist on joint motion. If the 'commanded' position would exceed a joint limit, DEPC will not output that portion of the command to the servos, and then DEPC will issue warning D6. The best method for recovery is for the external computer to issue a Re-start Command.

DEPC's response to the Interrupt Pushbutton is to issue warning D2.

DEPC's response to the Hold button is to complete the output in progress, and then wait until Hold Clear.

If the transmitter encounters a hardware problem, a warning D5 is issued.

* The two function bytes are the same as bytes two and three of the Point Data information. The function is executed when the robot reaches the coordinates that were received with the function.

2.4 Error Handling

A Warning instead of an Error is given first. If the response to the Warning is (CANCEL), then the P-IG Error occurs and the robot will enter the Pre-Manual Mode when the P-IG Error is canceled. If the response to the Warning is (ENTER), the robot re-starts DEPC.

2.5 Data Command

The byte information contained in a Data Command is identical to the information used by the External and/or the Adjust functions.

3.0 ENVIRONMENT

DEPC requires the SAT (CMI# 3-531-3502A) and SAR (CMI# 3-531-3503A) boards for operation of the RS232 link. These boards must have their jumpers set for the proper baud rate (must be greater than or equal to 9600 baud) and serial link protocol (RS232, RS422, or current loop). Cables (CMI# 3-422-1656A, and CMI# 3-422-2766A) are also required.

4.0 ARCHITECTURE

@DEPC (DEPC 93) is required for operation

@DXTR if both EXTERNAL and DEPC co-exist

@DPCF if only DEPC exists

@EXTR if only EXTERNAL function

@NXTR if neither DEPC nor EXTERNAL function exists

T3 Controller Error Messages:

Error Code	Problem
WD1	CTS low
WD2	Interrupt Button
WD3	Translation
WD4	Orientation
WD5	Trans hardware
WD6	Joint limit reached (Joint 6)
P05	Joint 2 reached upward DEPC limit
P0B	Joint limit reached
P9P	Wrist singularity problem
PIG	DEPC link terminated (unrecoverable)

APPENDIX A.1

Switch Settings

(Note: ON = CLOSED, OFF = OPEN)

TRANSMIT BOARD (SAT)

B (2)	OFF	ADDRESS
C (3)	OFF	
D (4)	OFF	
E (5)	OFF	
G (7)	OFF	INTERRUPTS
H	1-2	CRC 16
J	2-3	
K	2-3	
L	1-2	
R (1)	ON	1 STOP BIT
S (3)	ON	ASYNCH
T (4)	OFF	
A	1-2	
W (2)	ON	PARITY
X	1-2	RS232
Y	1-2	
Z	1-2	
TOGGLE SWITCH 1	UP	
BAUD RATE	14-15	(9600)

FOR NEWER VERSIONS OF SAT,
SW-F MUST BE CLOSED.

RECEIVE BOARD (SAR)

G (1)	OFF	INTERRUPTS
P	1-2	CRC 16
O	2-3	
N	2-3	
M	1-2	
B (8)	ON	TEST
K (6)	OFF	INTERRUPTS
J (5)	OFF	
H (4)	OFF	
C (2)	ON	ASYNCH
D (1)	ON	
A	1-2	
L (6)	OFF	PARITY
F (8)	ON	IPL DISABLED
R,S,T,U,V,W,X,Y		DON'T CARE

APPENDIX A.2

Interconnections

Put SAT board left of SAR board.

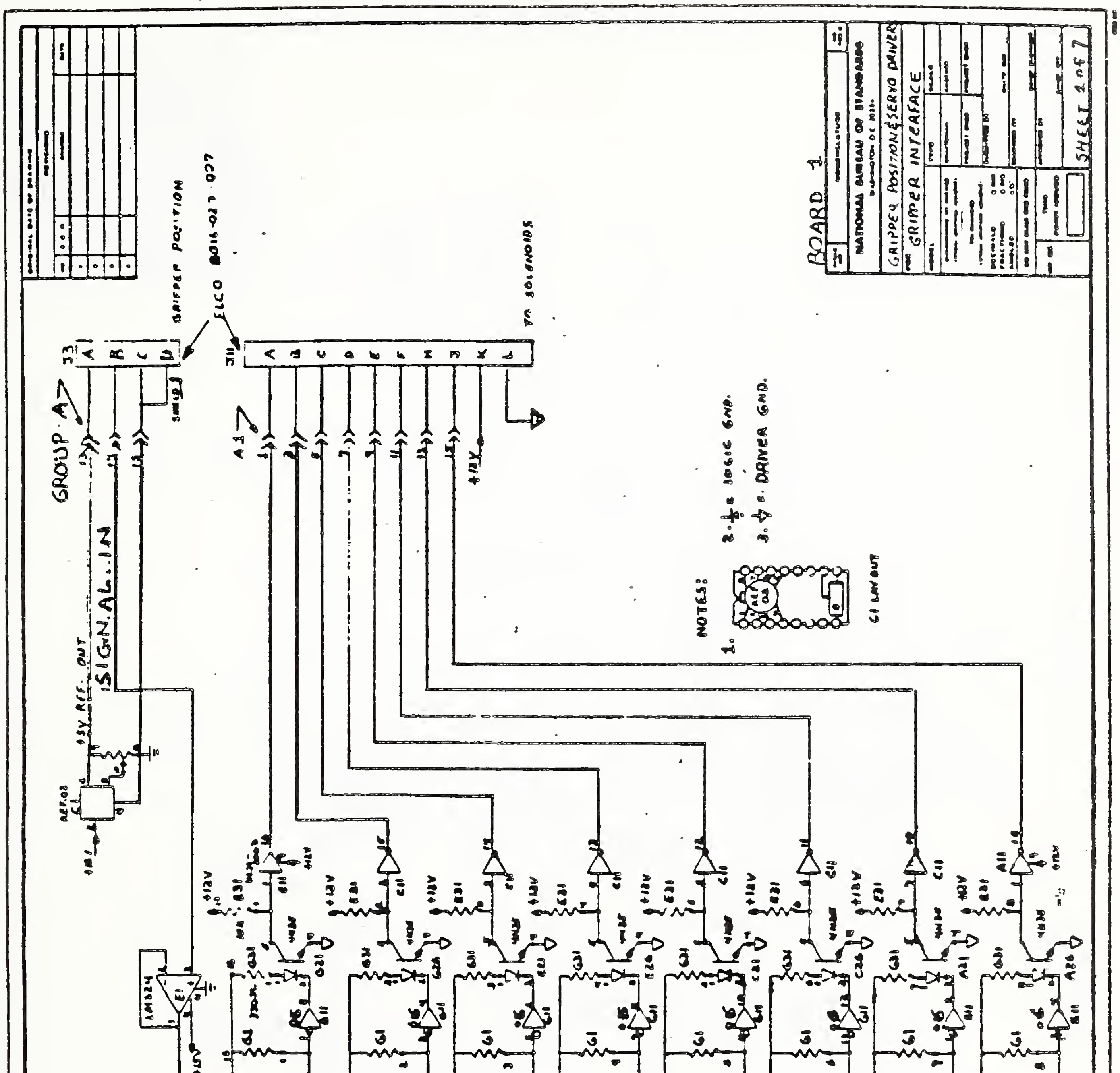
Connect top edge connectors with jumper cable.

Connect RS232 cable to SAT bottom edge connector.

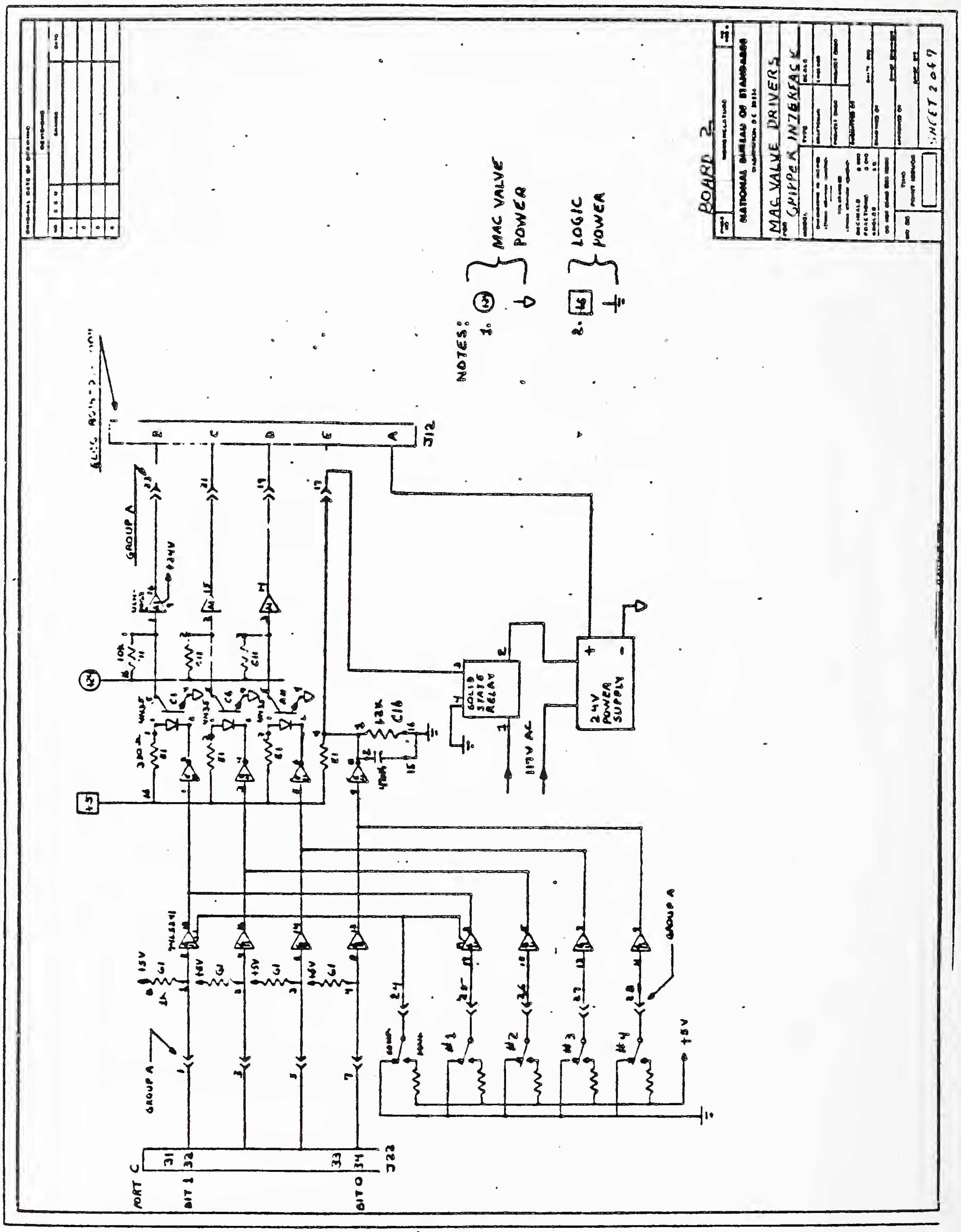
PIN 2	OUTPUT	ROBOT	XMIT	=	USER RECV
PIN 3	INPUT	ROBOT	RECV	=	USER XMIT
PIN 7	GROUND				
PIN 20	OUTPUT	ROBOT	DTR	=	USER CTS
PIN 5	INPUT	ROBOT	CTS	=	USER DTR

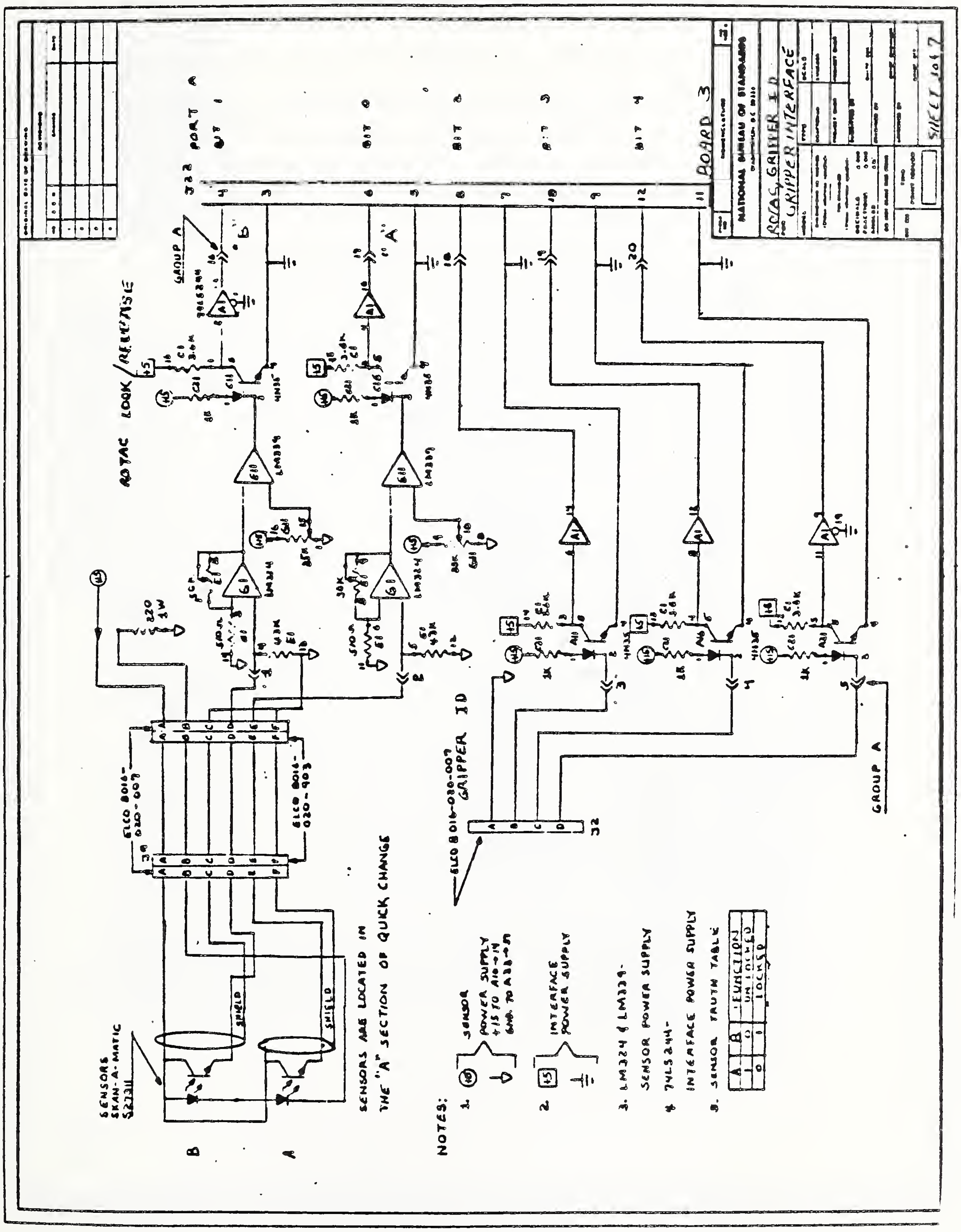
APPENDIX B

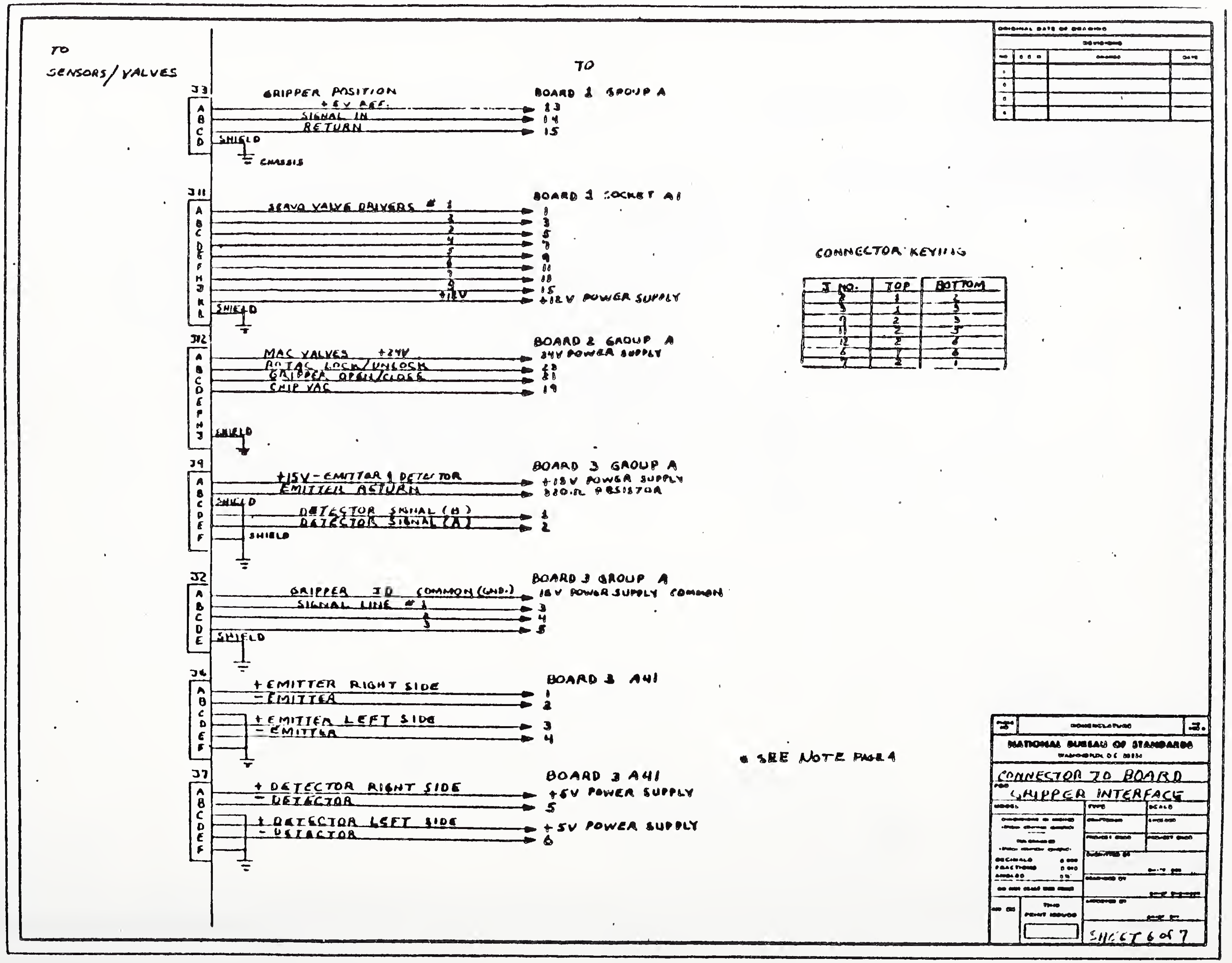
ELECTRICAL SCHEMATICS

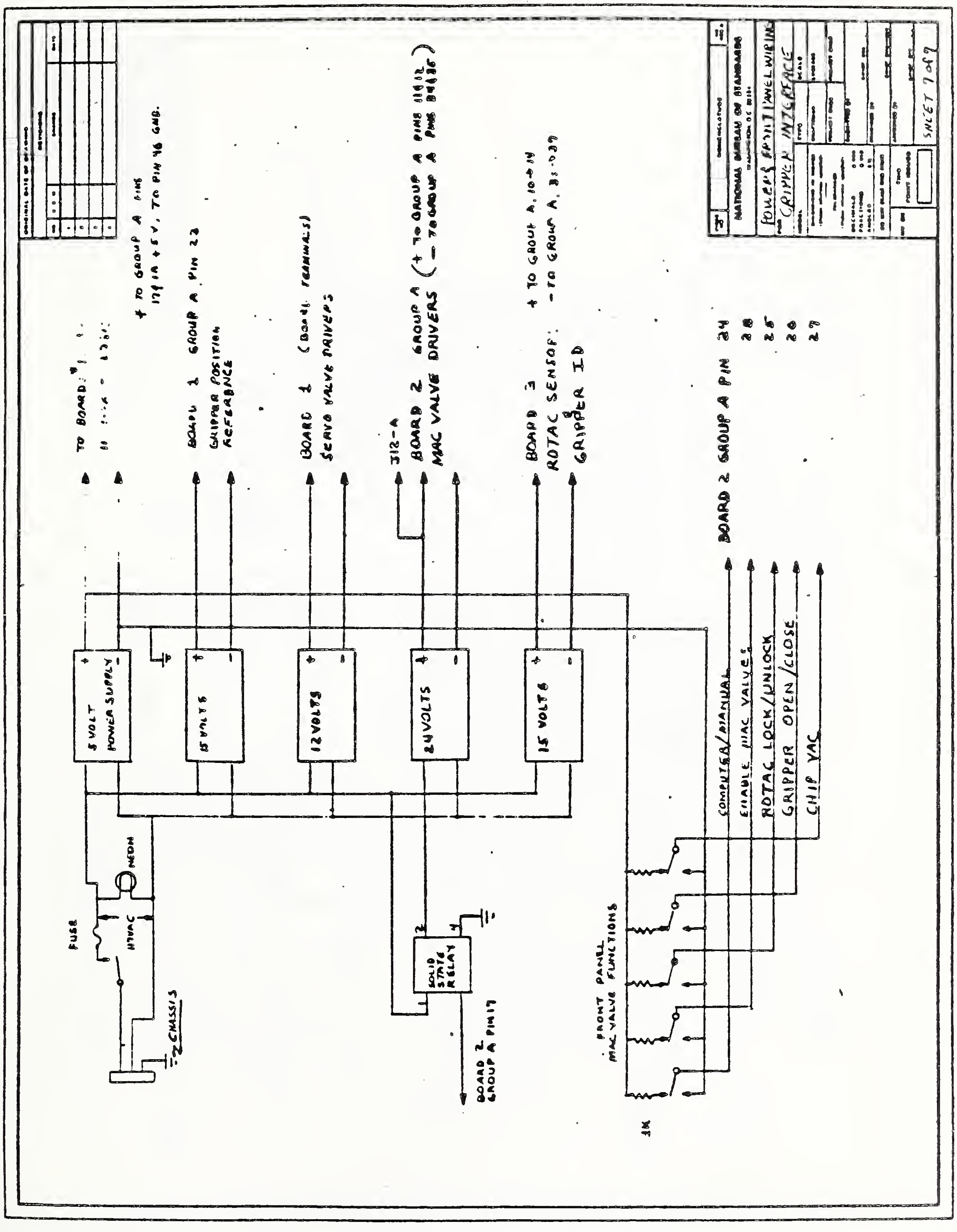


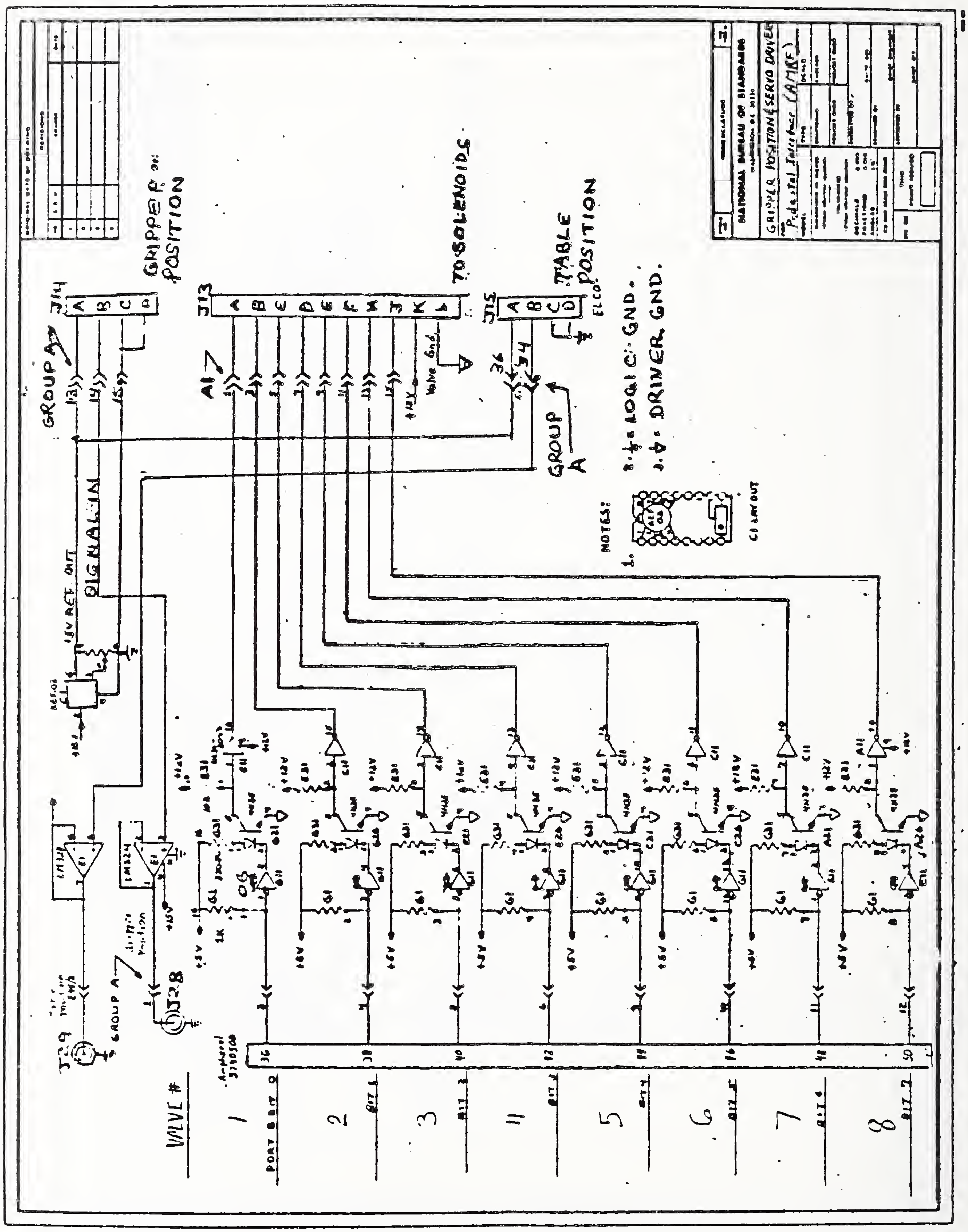
BOARD 1
NATIONAL BUREAU OF STANDARDS
GRIPPER POSITION & SERVO DRIVER
GRIPPER INTERFACE
SHEET 1 of 7
GROUP A
SIGNAL IN
GRIPPER POSITION
TO SOLENOIDS
NOTES:
LOGIC GND.
DRIVER GND.
C1 LAYOUT
LM324
J3
J11
J22
J24


BOARD 2
NATIONAL BUREAU OF STANDARDS
MAC VALVE DRIVERS
GRIPPER INTERFACE
NOTES:
1. 24 MAC VALVE POWER
2. +5 LOGIC POWER
SOLID STATE RELAY
24V POWER SUPPLY
117V AC
GROUP A
PORT C
BIT 1
BIT 0
J22
J12
+5V

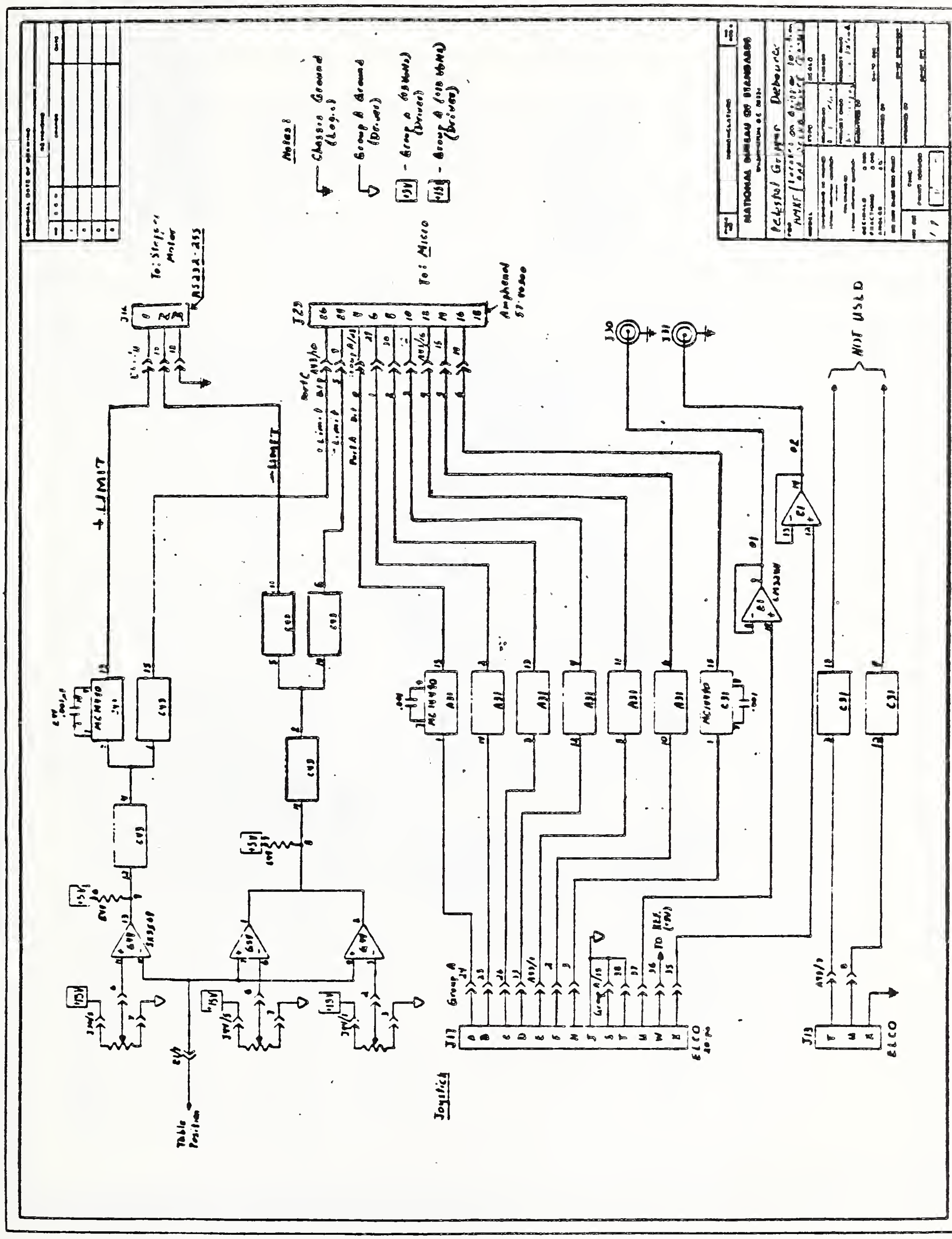

ROTAC LOOK/RELEASE
PORT A
GROUP A
GRIPPER ID
SENSORS SKAN-A-MATIC S27311
SENSORS ARE LOCATED IN THE "A" SECTION OF QUICK CHANGE
NOTES:
1. SENSOR POWER SUPPLY
2. INTERFACE POWER SUPPLY
3. LM324 & LM339 - SENSOR POWER SUPPLY
4. 74LS244 - INTERFACE POWER SUPPLY
5. SENSOR TRUTH TABLE
NATIONAL BUREAU OF STANDARDS
ROTAC GRIPPER I.D. GRIPPER INTERFACE
BOARD 3
SHEET 1 of 7


TO SENSORS/VALVES
TO
J3
GRIPPER POSITION
+5V REF.
SIGNAL IN
RETURN
SHIELD
CHASSIS
BOARD 1 GROUP A
13
14
15
J11
SERVO VALVE DRIVERS # 1
+12V
SHIELD
BOARD 1 SOCKET A1
+12 V POWER SUPPLY
J12
MAC VALVES +24V
ROTAC LOCK/UNLOCK
GRIPPER OPEN/CLOSE
CHIP VAC
SHIELD
BOARD 2 GROUP A
24V POWER SUPPLY
J9
+15V - EMITTER & DETECTOR
EMITTER RETURN
SHIELD
DETECTOR SIGNAL (B)
DETECTOR SIGNAL (A)
BOARD 3 GROUP A
+15V POWER SUPPLY
330 Ω RESISTOR
J2
GRIPPER ID COMMON (GND.)
SIGNAL LINE # 1
SHIELD
BOARD 3 GROUP A
15V POWER SUPPLY COMMON
J6
+ EMITTER RIGHT SIDE
- EMITTER
+ EMITTER LEFT SIDE
- EMITTER
BOARD 3 A41
J7
+ DETECTOR RIGHT SIDE
- DETECTOR
+ DETECTOR LEFT SIDE
- DETECTOR
BOARD 3 A41
+5V POWER SUPPLY
+5V POWER SUPPLY
CONNECTOR KEYING
J NO.
TOP
BOTTOM
* SEE NOTE PAGE 4
NATIONAL BUREAU OF STANDARDS
CONNECTOR TO BOARD
GRIPPER INTERFACE
SHEET 6 of 7

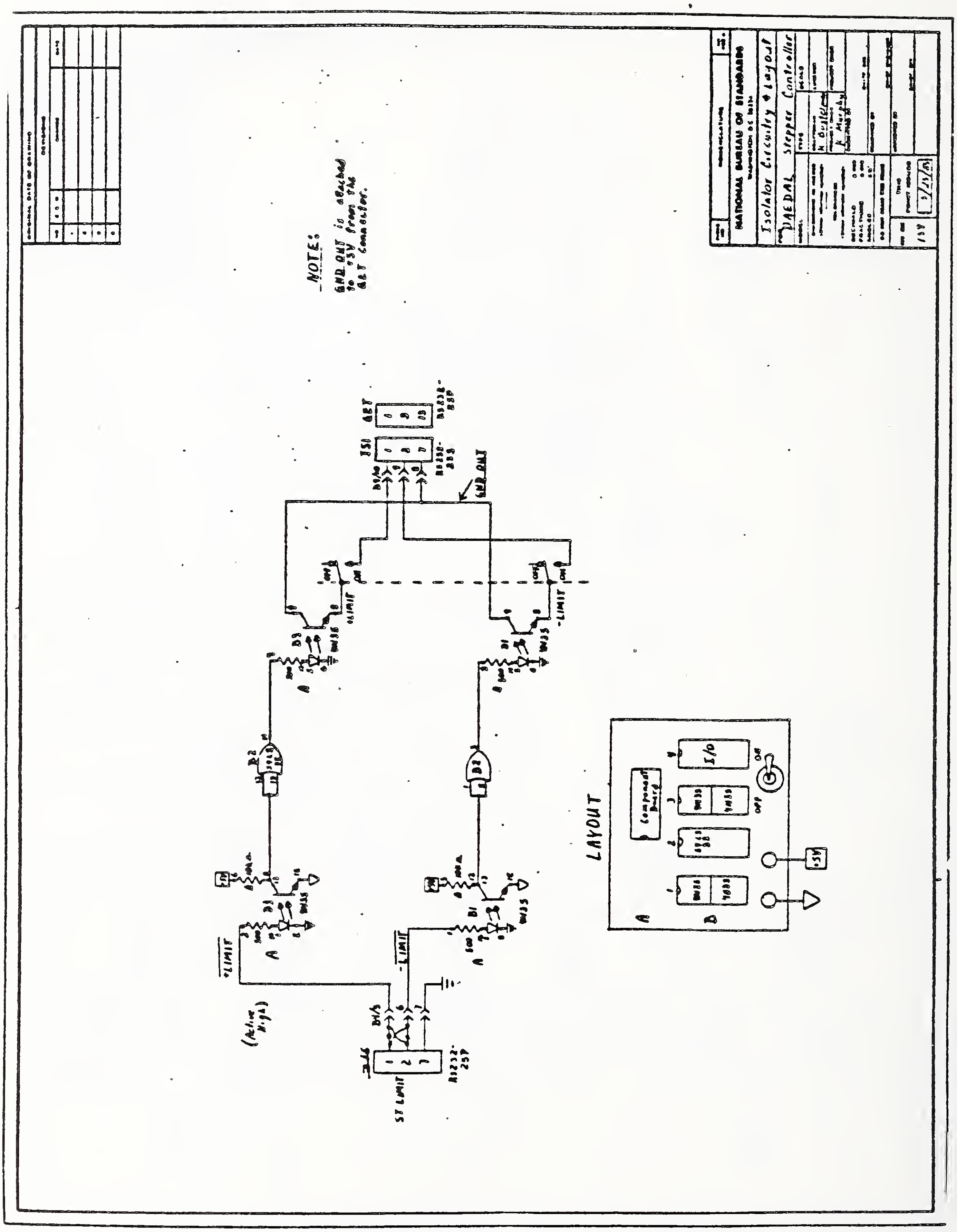

TO BOARD 1
BOARD 1 GROUP A PIN 22
GRIPPER POSITION REFERENCE
BOARD 1 (BOARD TERMINALS)
SERVO VALVE DRIVERS
J12-A
BOARD 2 GROUP A
MAC VALVE DRIVERS
BOARD 3
ROTAC SENSOR
GRIPPER ID
BOARD 2 GROUP A PIN 24
28
25
26
27
5 VOLT POWER SUPPLY
15 VOLTS
12 VOLTS
24 VOLTS
15 VOLTS
COMPUTER/MANUAL
ENABLE MAC VALVES
ROTAC LOCK/UNLOCK
GRIPPER OPEN/CLOSE
CHIP VAC
FUSE
117VAC
NEON
CHASSIS
SOLID STATE RELAY
BOARD 2 GROUP A PIN 17
FRONT PANEL MAC VALVE FUNCTIONS
1K
NATIONAL BUREAU OF STANDARDS
POWER & FRONT PANEL WIRING
GRIPPER INTERFACE
SHEET 7 of 9

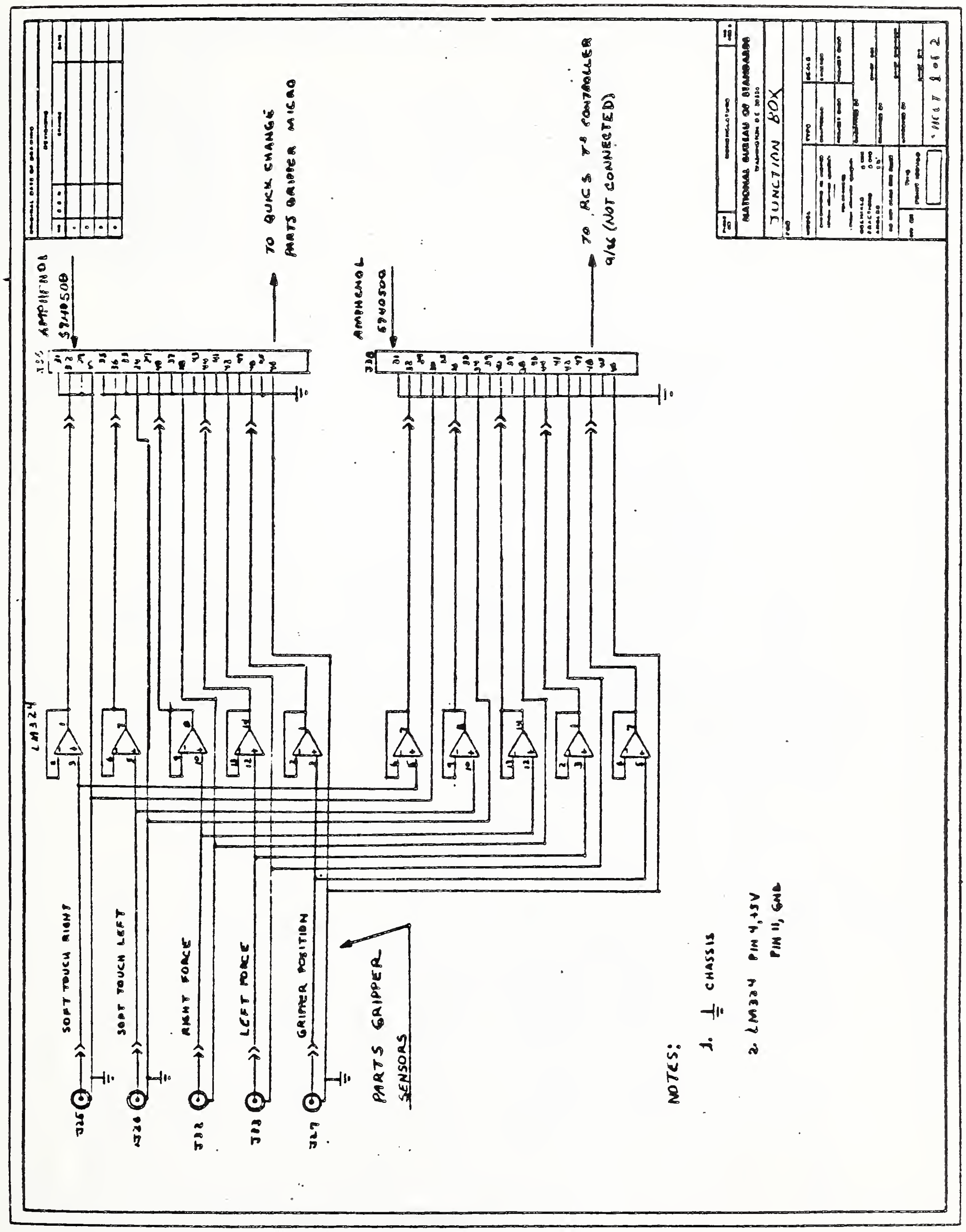


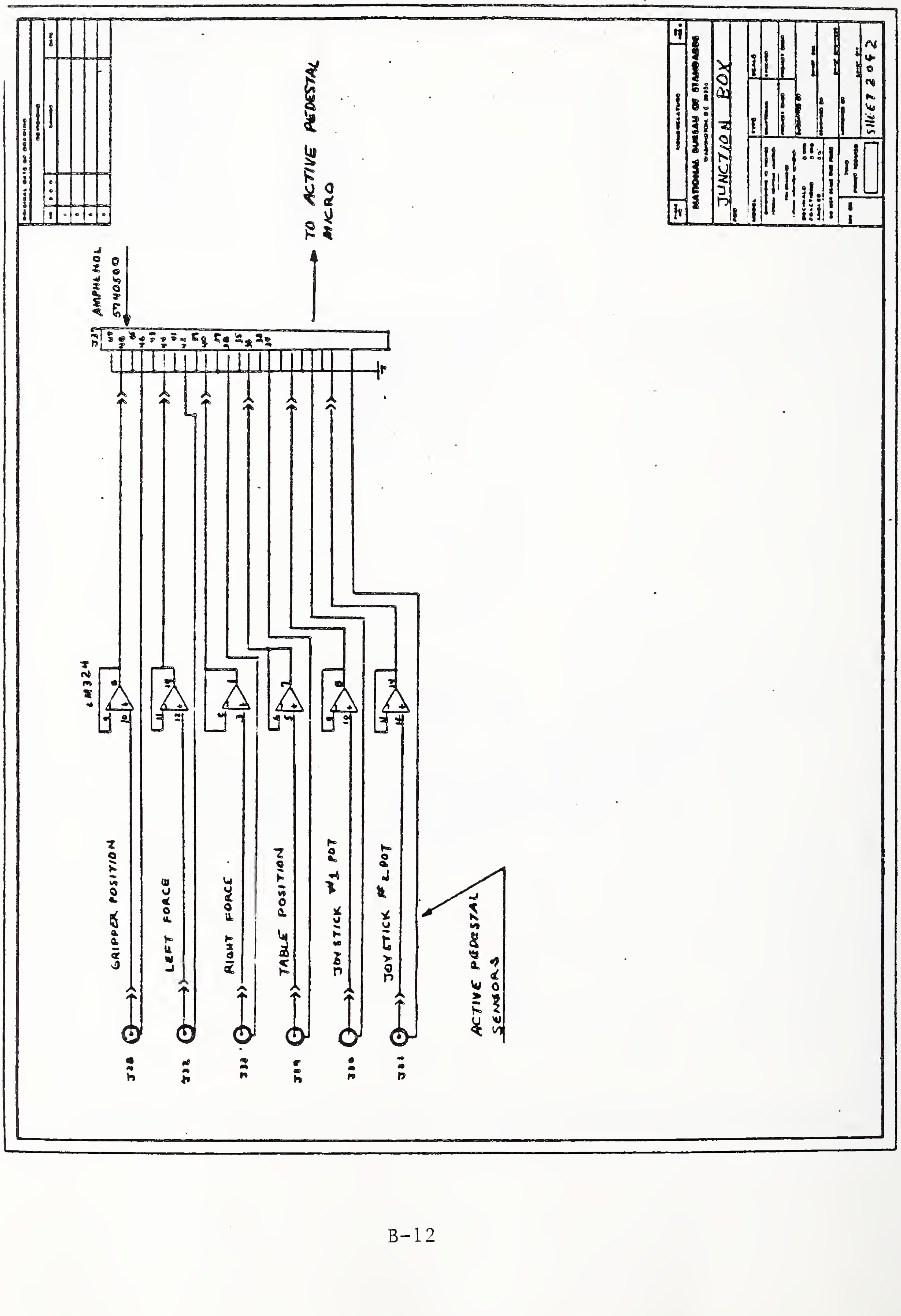
VALVE #
GRIPPER POSITION
TO SOLENOIDS
TABLE POSITION
GROUP A
SIGNAL IN
NOTES:
LOGIC GND.
DRIVER GND.
NATIONAL BUREAU OF STANDARDS
GRIPPER POSITION & SERVO DRIVER
Pedestal Interface (AMRF)



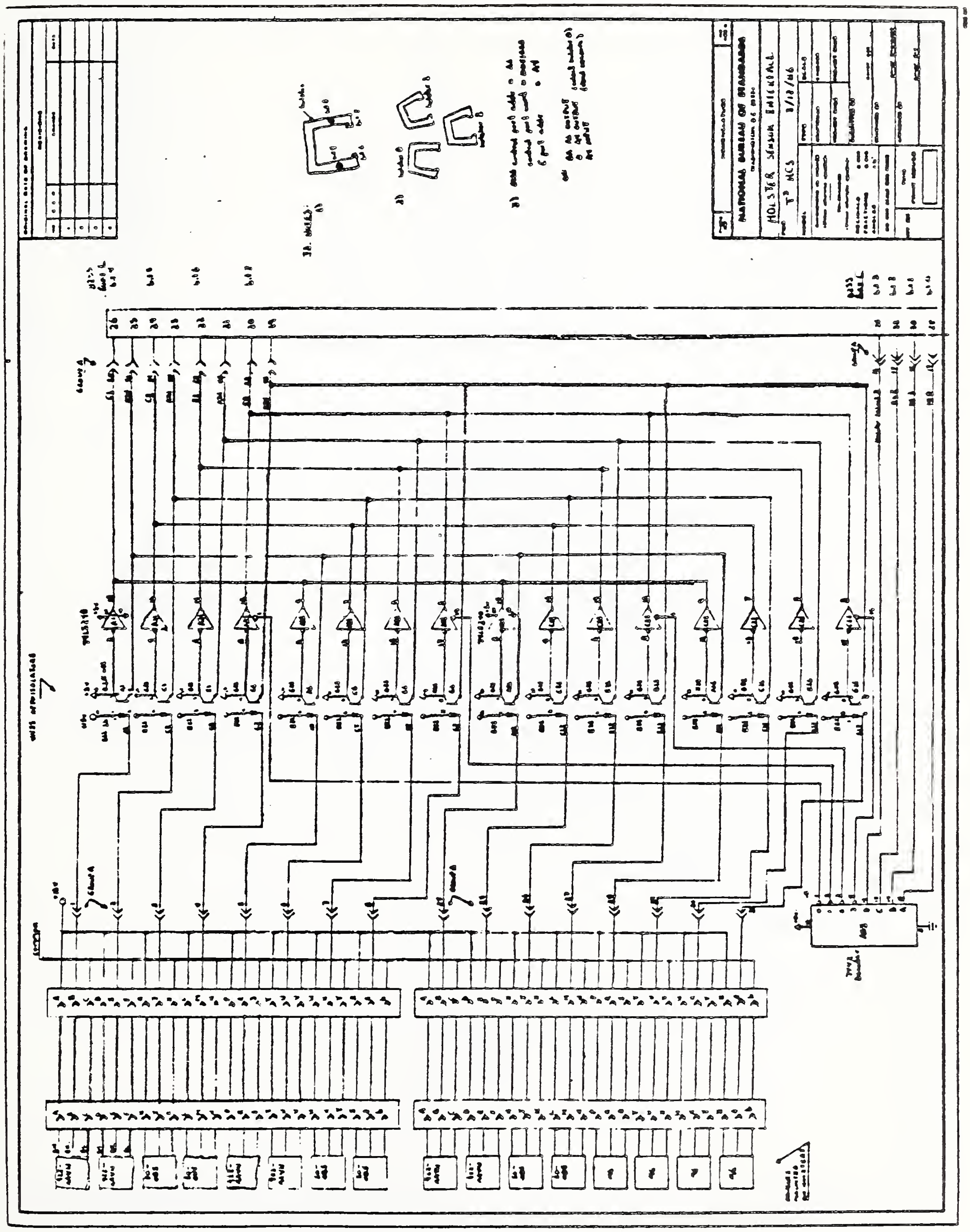
+LIMIT
-LIMIT
NOT USED
Notes:
Joystick
Table Position
To: Micro
ELCO


NATIONAL BUREAU OF STANDARDS
Isolator Circuitry + Layout
DAEDAL Stepper Controller
NOTES:
GND OUT is attached to +5V from the ART connector.
GND OUT
LAYOUT
Component Board
I/O
+LIMIT
-LIMIT
(Active High)

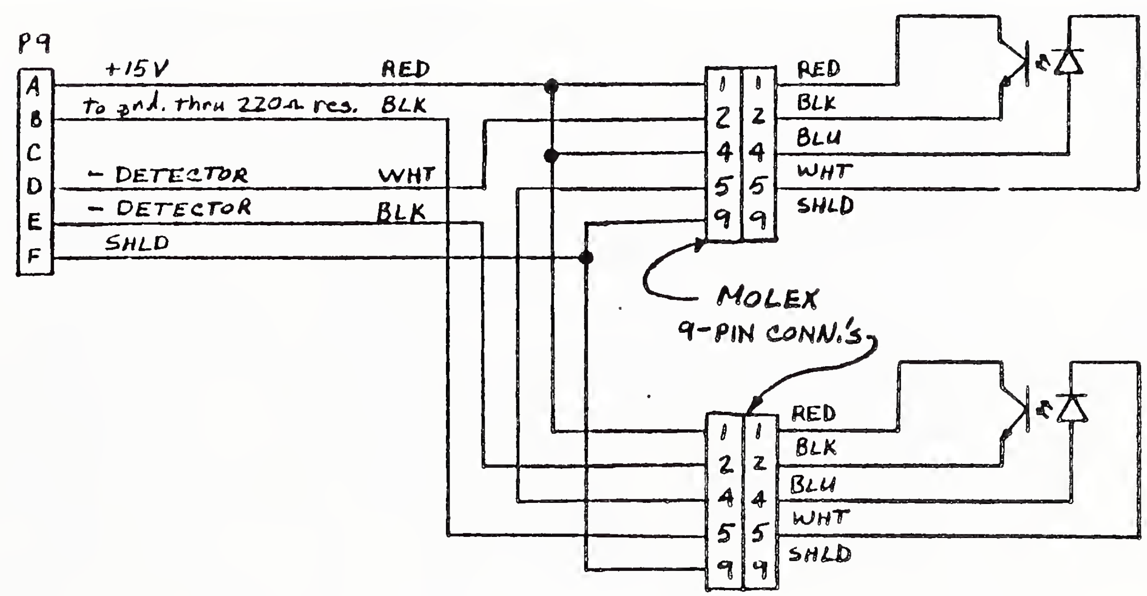

NATIONAL BUREAU OF STANDARDS
JUNCTION BOX
TO QUICK CHANGE PARTS GRIPPER MICRO
TO RCS T8 CONTROLLER
(NOT CONNECTED)
LM324
SOFT TOUCH RIGHT
SOFT TOUCH LEFT
RIGHT FORCE
LEFT FORCE
GRIPPER POSITION
PARTS GRIPPER SENSORS
NOTES:
1. CHASSIS
2. LM324 PIN 4, +5V
PIN 11, GND



TO ACTIVE PEDESTAL MICRO
AMPHENOL
LM324
GRIPPER POSITION
LEFT FORCE
RIGHT FORCE
TABLE POSITION
ACTIVE PEDESTAL SENSORS
NATIONAL BUREAU OF STANDARDS
JUNCTION BOX



MOLEX CONNECTORS & OPTICAL PROX. SENSORS INSIDE Q.C ("A" PLATE)

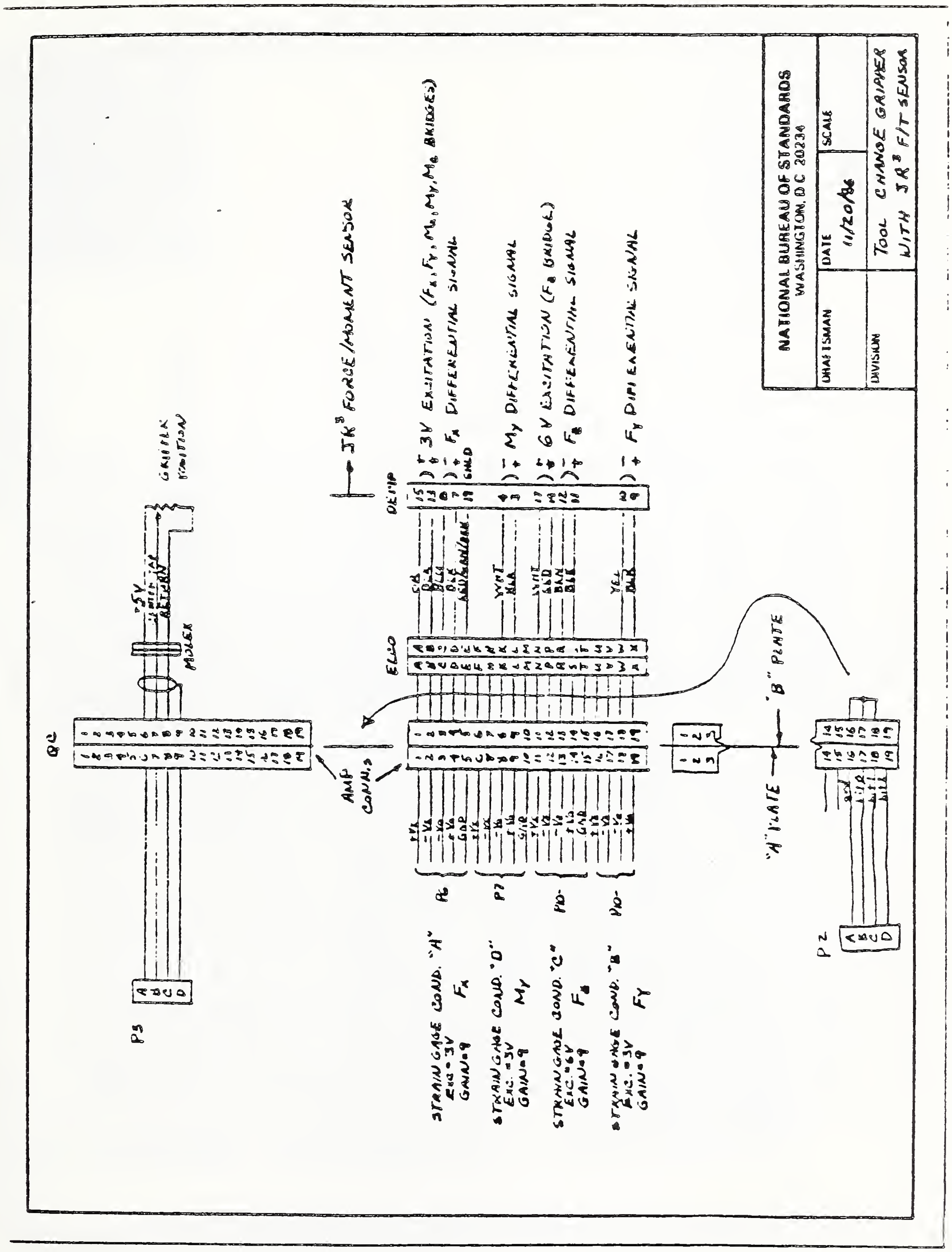


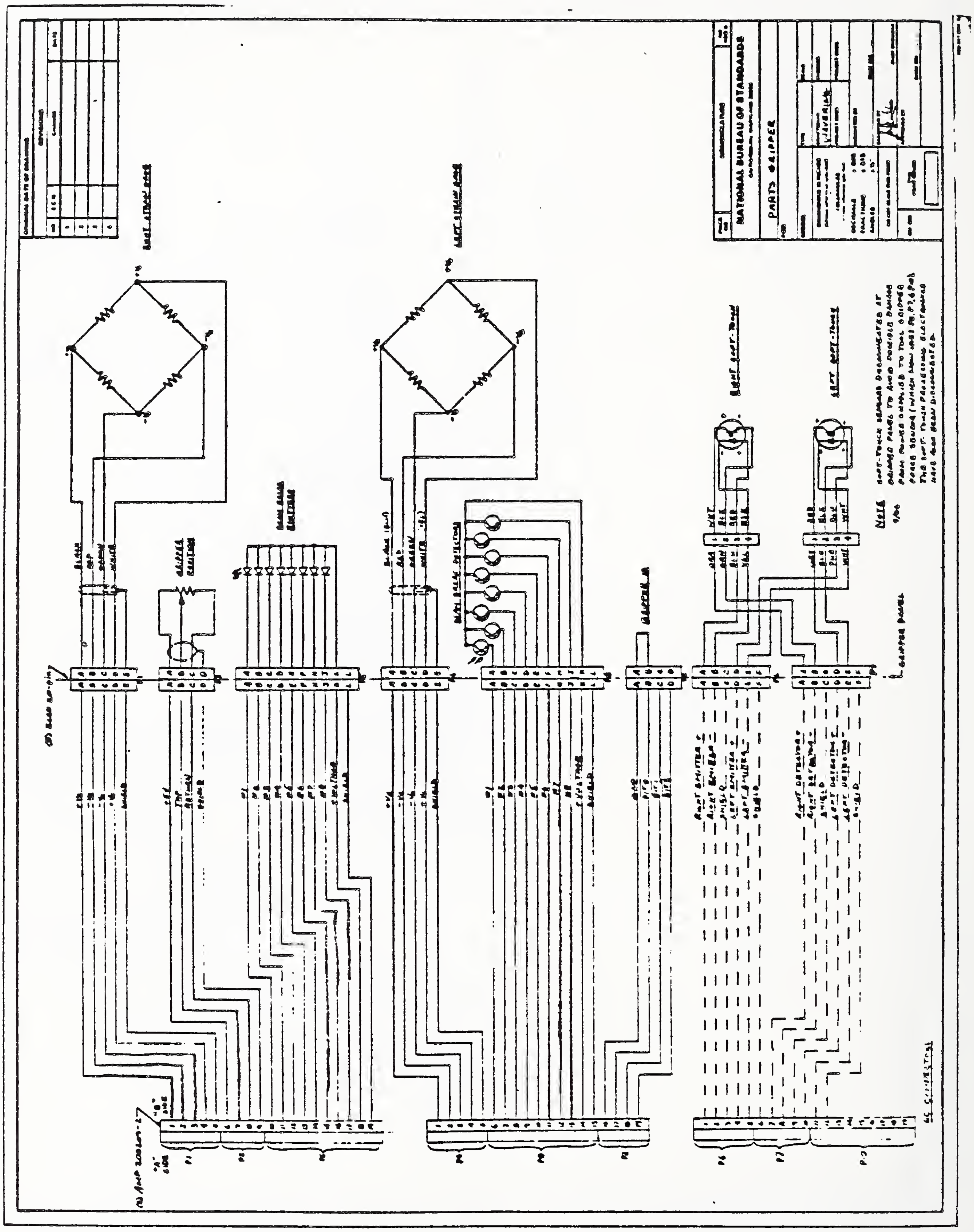
— Add to gripper interface —

NATIONAL BUREAU OF STANDARDS
WASHINGTON, D.C. 20234

DRAFTSMAN	DATE	SCALE
DIVISION	QC ROTAC SENSORS	

NBS 650 (Rev 10 83)


NATIONAL BUREAU OF STANDARDS
WASHINGTON, D.C. 20234
DRAFTSMAN
DATE
SCALE
DIVISION
TOOL CHANGE GRIPPER WITH JR3 F/T SENSOR
JR3 FORCE/MOMENT SENSOR
QC
P3
P2
AMP CONN'S
"A" PLATE
"B" PLATE
MOLEX
GRIPPER POSITION
ELCO
STRAIN GAGE COND. "A"
STRAIN GAGE COND. "D"
STRAIN GAGE COND. "C"
STRAIN GAGE COND. "B"
My DIFFERENTIAL SIGNAL


NATIONAL BUREAU OF STANDARDS
PARTS GRIPPER

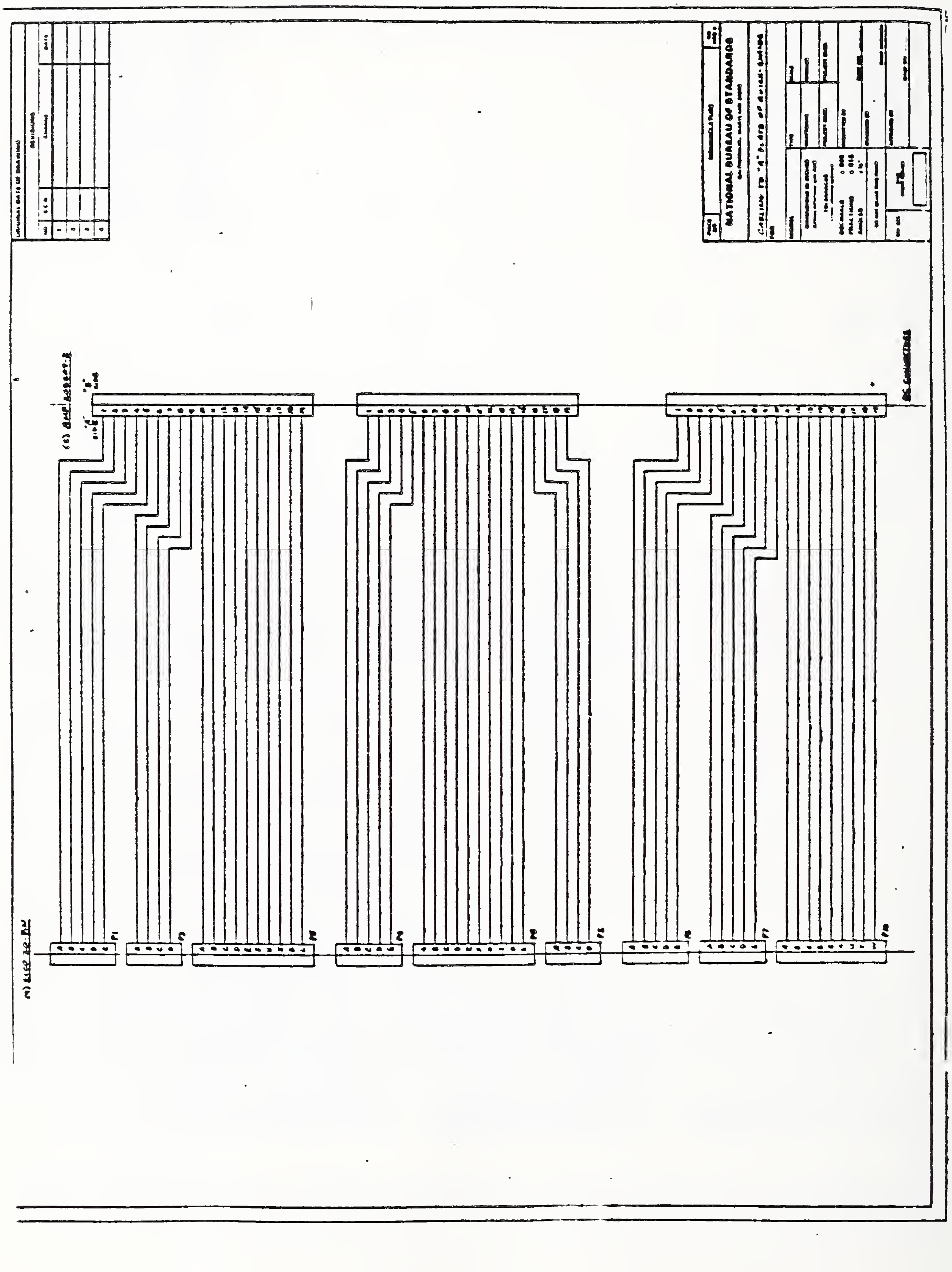
J NO	FROM	TO	FUNCTION	Connector Type	KEYING TOP	KEYING BOTTOM
1	Strain Gage Cond.	J1 T³	Force Sensor Rt Side	BENDIX PT00A-18-10P ELCO-20P	1	1
2	Gripper Interface	J2 T³	Gripper ID	ELCO-20P	1	2
3	Gripper Interface	J3 T³	Gripper Position	ELCO 20P	1	3
4	Strain Gage Cond.	J4 T³	Force Sensor Lft. Side	BENDIX PT00A-18-10P ELCO-20P	1	4
5	Beam Break Chassis	J5 T³	Beam Break Emitter	ELCO-20P	1	5
6	Gripper Interface	J6 T³	Soft Touch Emitters	ELCO-20P	1	6
7	Gripper Interface	J7 T³	Soft Touch Detectors	ELCO-20P	2	1
8	Beam Break Chassis	J8 T³	Beam Break Detectors	ELCO-20P	2	2
9	Gripper Interface	J9 T³	Tool Lock/Unlock	ELCO-20P	2	3
10	—	—	—	—		
11	Gripper Interface	J11 T³	Servo drivers	ELCO-20P	2	5
12	Gripper Interface	J12 T³	Mac Valve Driver	ELCO-20P	2	6
13	Pedestal Interface	Active Pedestal	Servo drivers	ELCO-20P	3	1
14	Pedestal Interface	Active Pedestal	Gripper Position	ELCO-20P	3	2
15	Pedestal Interface	Active Pedestal	Table Position	ELCO-20P	3	3
16	Pedestal Interface	STEPPER CONTROL	Table Limits	RS232-25P		
17	Pedestal Interface	Joystick	Active Ped. Control	ELCO-20P	3	4
18	Strain Gage Cond.	Active Pedestal	Force Sensor Rt.	BENDIX PT00A-12-10P ELCO-20P	3	6
19	Strain Gage Cond.	Active Pedestal	Force Sensor Lft.	BENDIX PT00A-18-10P ELCO-20P	4	1
20	Stepper Control	Active Pedestal	Motor Drive	JONES		
21	Beam Break Chassis	Gripper Holster	Holster Detectors	ELCO-56P	3	5
22	Gripper Interface	Multi-Bus Bucket	I/O	3M 3368 3M 3418		
23	Pedestal Interface	Multi-Bus Bucket	I/O	3M 3368 3M 3418		
24	Beam Break Chassis	T³ Controller (RCS)	I/O	3M 3368 3M 3418		
25	Gripper Interface	Analog Interface	Soft Touch Rt.	BNC		
26	Gripper Interface	Analog Interface	Soft Touch Lft.	BNC		
27	Gripper Interface	Analog Interface	Gripper Position	BNC		
28	Pedestal Interface	Analog Interface	Gripper Position	BNC		
29	Pedestal Interface	Analog Interface	Table Position	BNC		
30	Pedestal Interface	Analog Interface	Joystick Pot #1	BNC		
31	Pedestal Interface	Analog Interface	Joystick Pot #2	BNC		
32	Strain Gage Cond.	Analog Interface	Right Force	JONES BNC	1	
33	Strain Gage Cond.	Analog Interface	Left Force	JONES BNC		
34	Strain Gage Cond.	Analog Interface	Right Force	JONES BNC		
35	Strain Gage Cond.	Analog Interface	Left Force	JONES BNC		
36	Analog Interface	Quick Change Micro	Analog Signal	3M 3368 3M 3418		
37	Analog Interface	Active Ped. Micro	Analog Signal	3M 3368 3M 3418		
38	Analog Interface	RCS	Analog Signal	3M 3368 3M 3418		

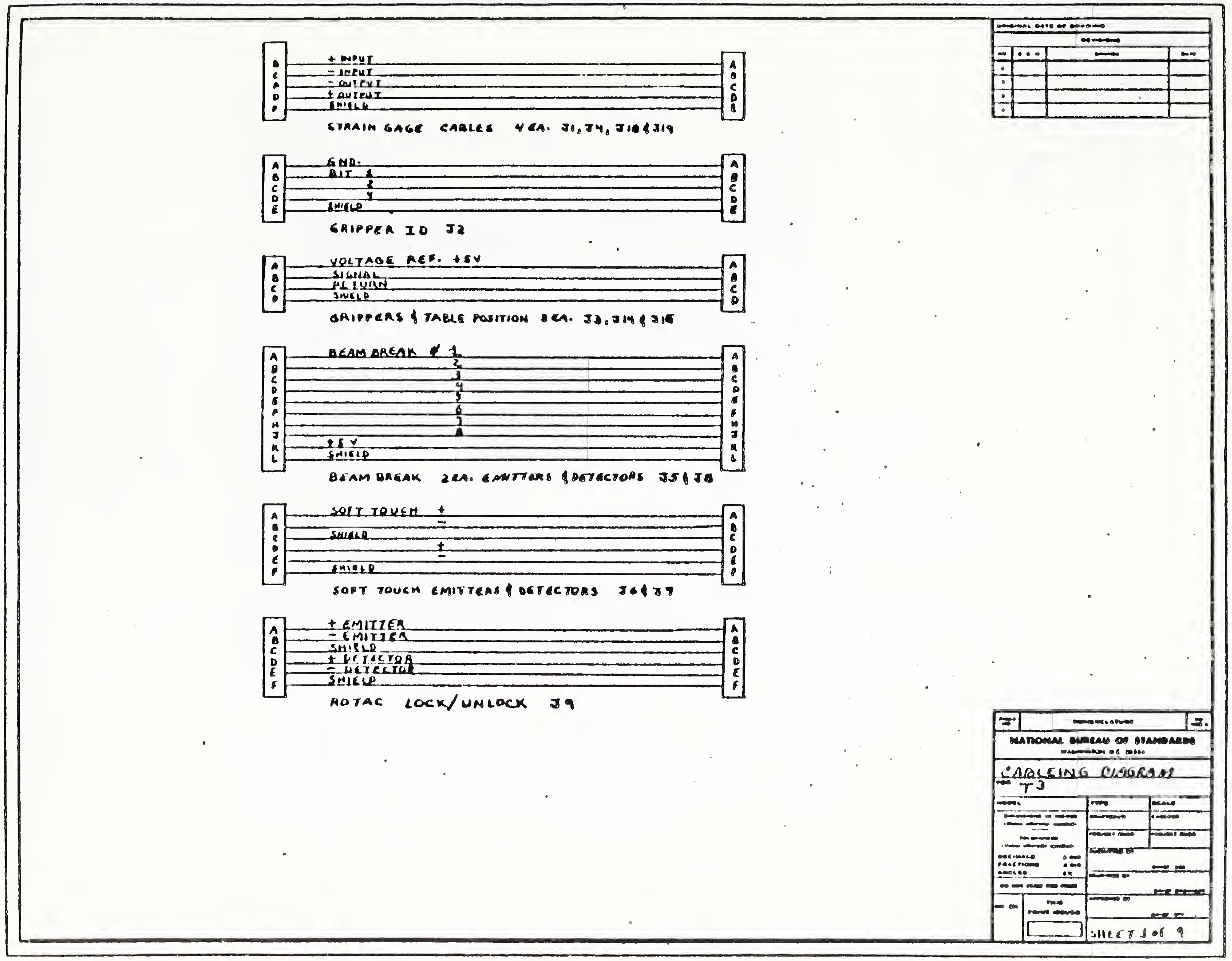
NATIONAL BUREAU OF STANDARDS

CABLE LIST

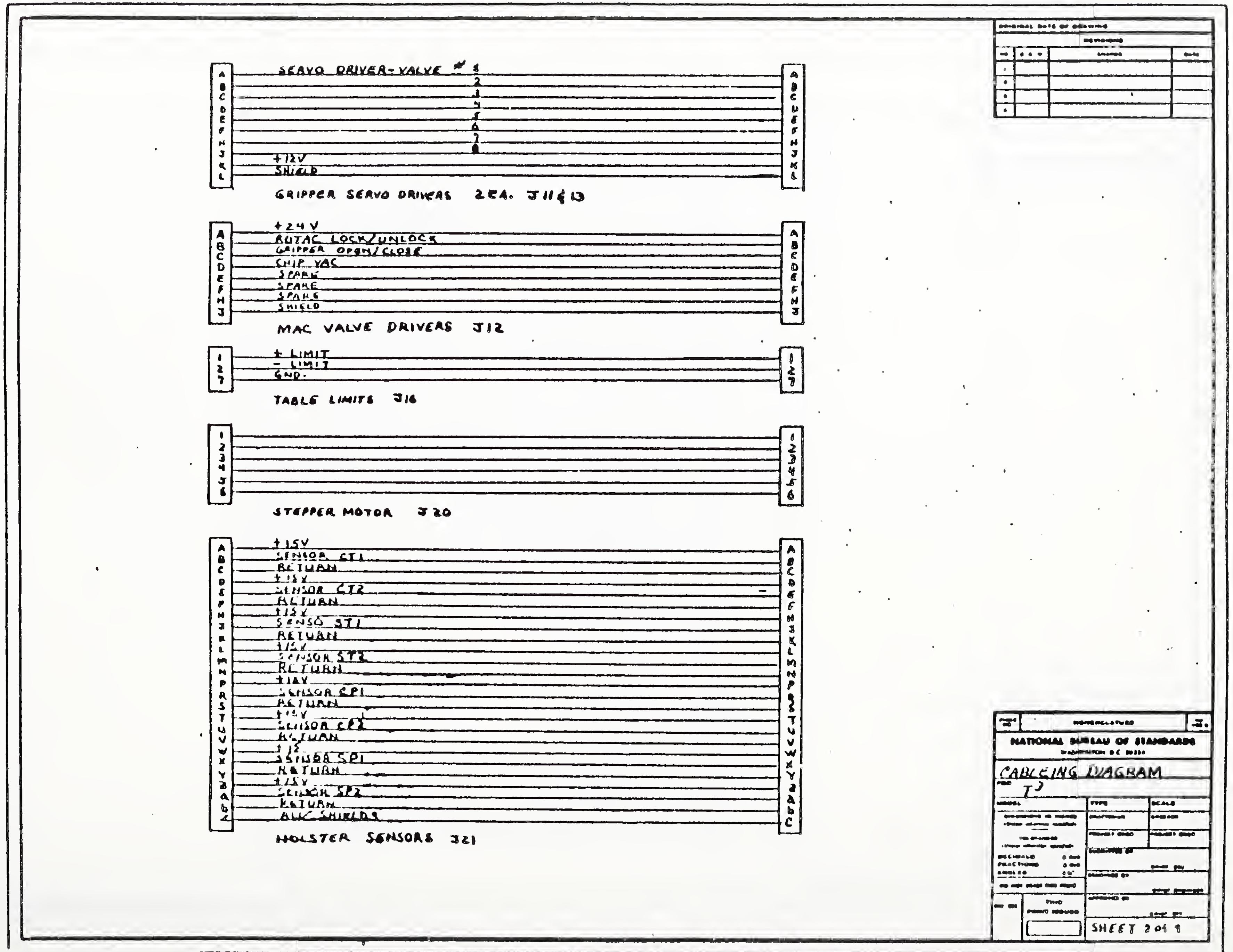
AMRF

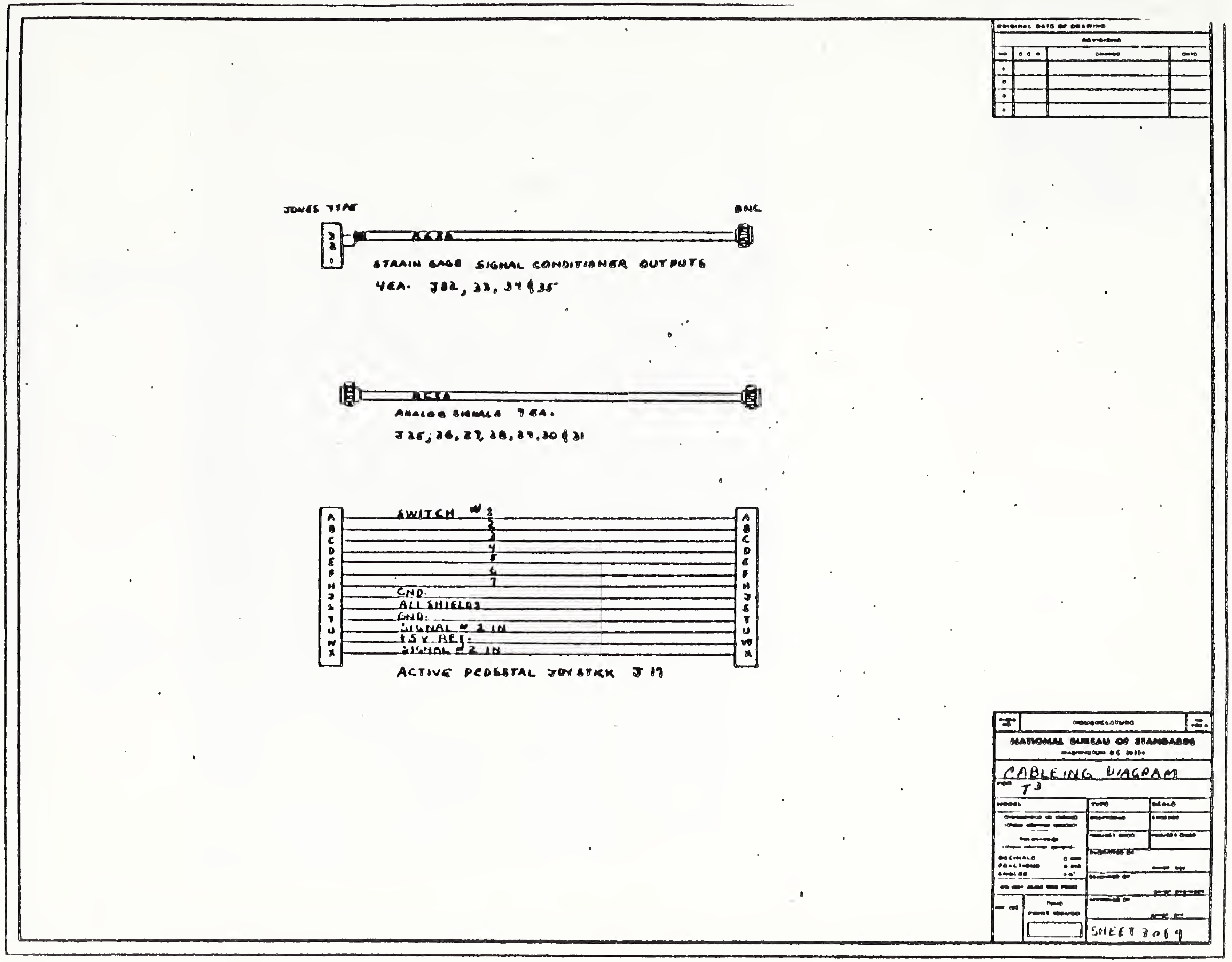
7/19/82

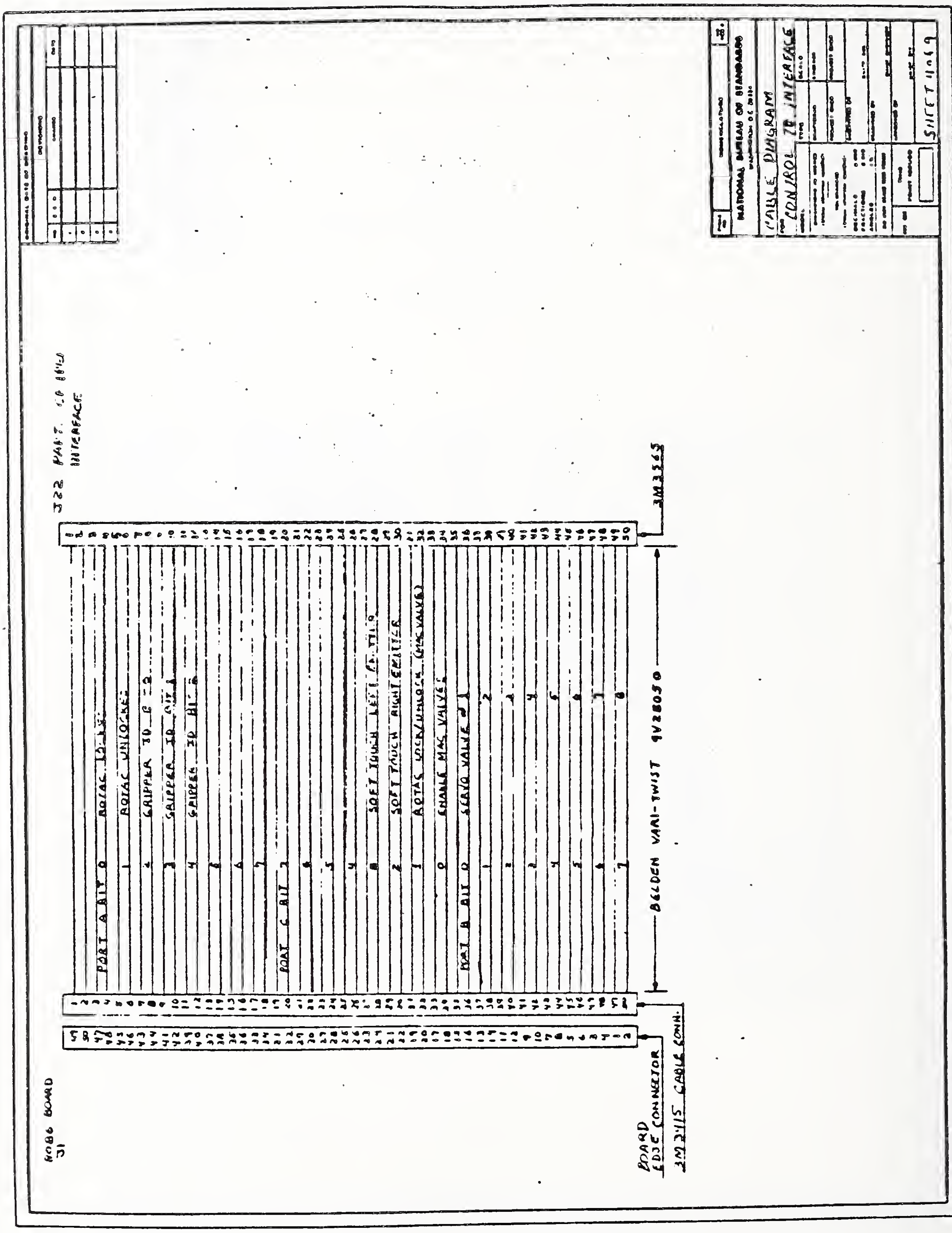

NATIONAL BUREAU OF STANDARDS

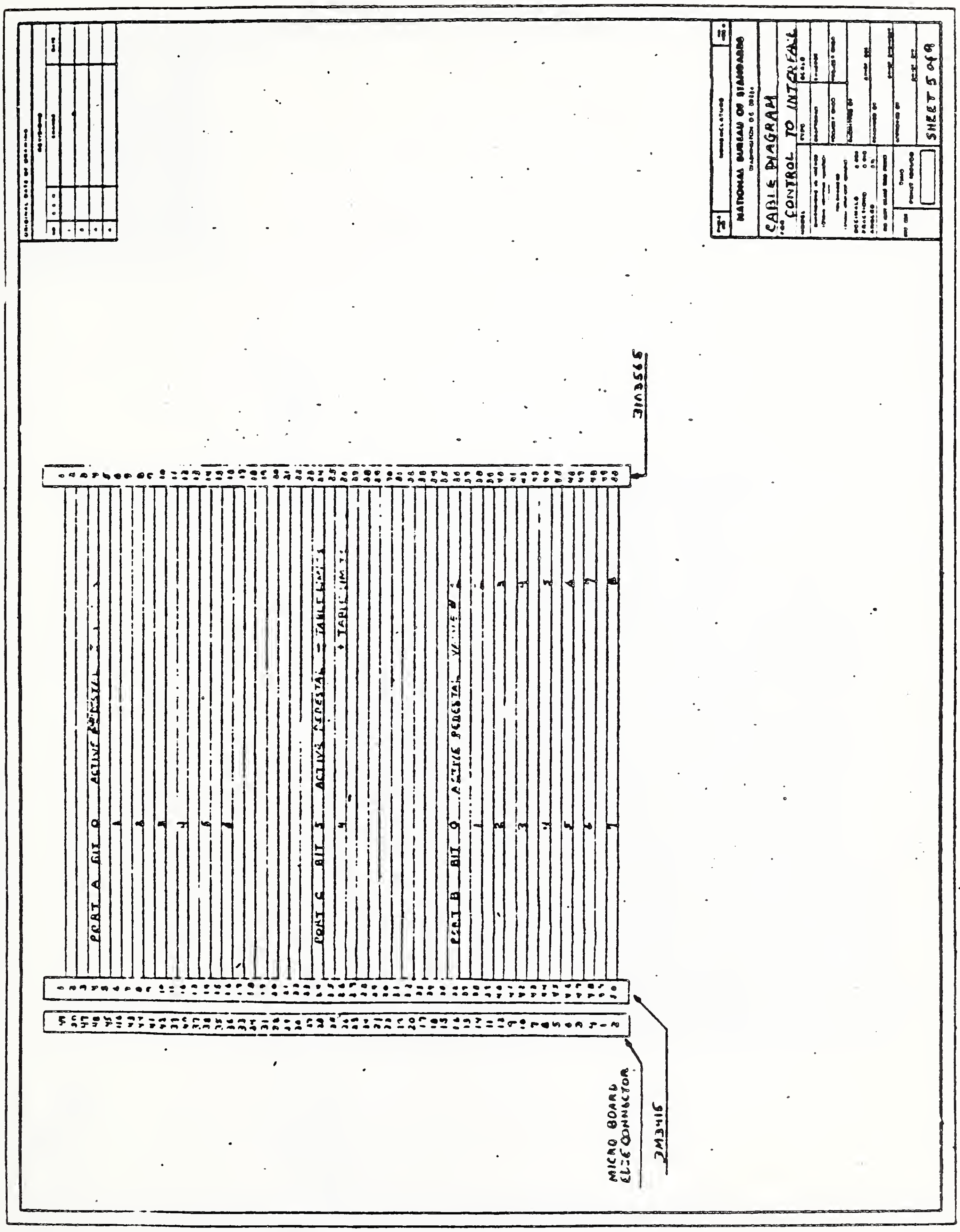


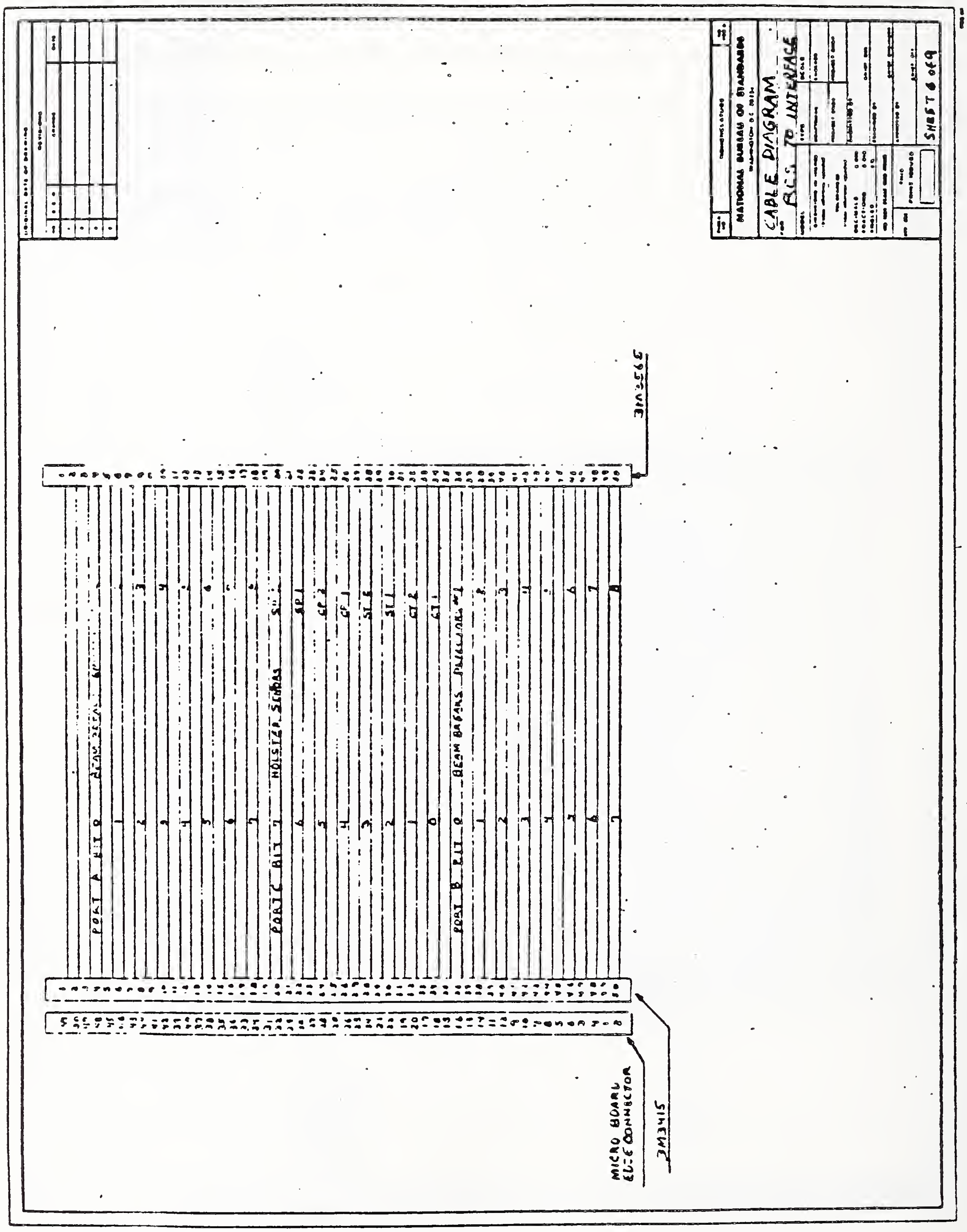
+ INPUT
− INPUT
− OUTPUT
+ OUTPUT
SHIELD
STRAIN GAGE CABLES 4 EA. J1, J4, J18 & J19
GND.
BIT 1
2
4
SHIELD
GRIPPER ID J2
VOLTAGE REF. +5V
SIGNAL
RETURN
SHIELD
GRIPPERS & TABLE POSITION 3 EA. J3, J14 & J15
BEAM BREAK # 1
2
3
4
5
6
7
8
+5 V
SHIELD
BEAM BREAK 2 EA. EMITTERS & DETECTORS J5 & J8
SOFT TOUCH +
−
SHIELD
+
−
SHIELD
SOFT TOUCH EMITTERS & DETECTORS J6 & J7
+ EMITTER
− EMITTER
SHIELD
+ DETECTOR
− DETECTOR
SHIELD
ROTAC LOCK/UNLOCK J9
NATIONAL BUREAU OF STANDARDS
CABLEING DIAGRAM
T3
SHEET 1 of 9


SERVO DRIVER-VALVE # 1
+12V
SHIELD
GRIPPER SERVO DRIVERS 2EA. J11 & 13
+24V
ROTAC LOCK/UNLOCK
GRIPPER OPEN/CLOSE
CHIP VAC
SPARE
SPARE
SPARE
SHIELD
MAC VALVE DRIVERS J12
+ LIMIT
- LIMIT
GND.
TABLE LIMITS J16
STEPPER MOTOR J20
+15V
SENSOR CT1
RETURN
+15V
SENSOR CT2
RETURN
+15V
SENSO ST1
RETURN
+15V
SENSOR ST2
RETURN
+15V
SENSOR CP1
RETURN
+15V
SENSOR CP2
RETURN
+15V
SENSOR SP1
RETURN
+15V
SENSOR SP2
RETURN
ALL SHIELDS
HOLSTER SENSORS J21
NATIONAL BUREAU OF STANDARDS
CABLEING DIAGRAM
SHEET 2 of 9

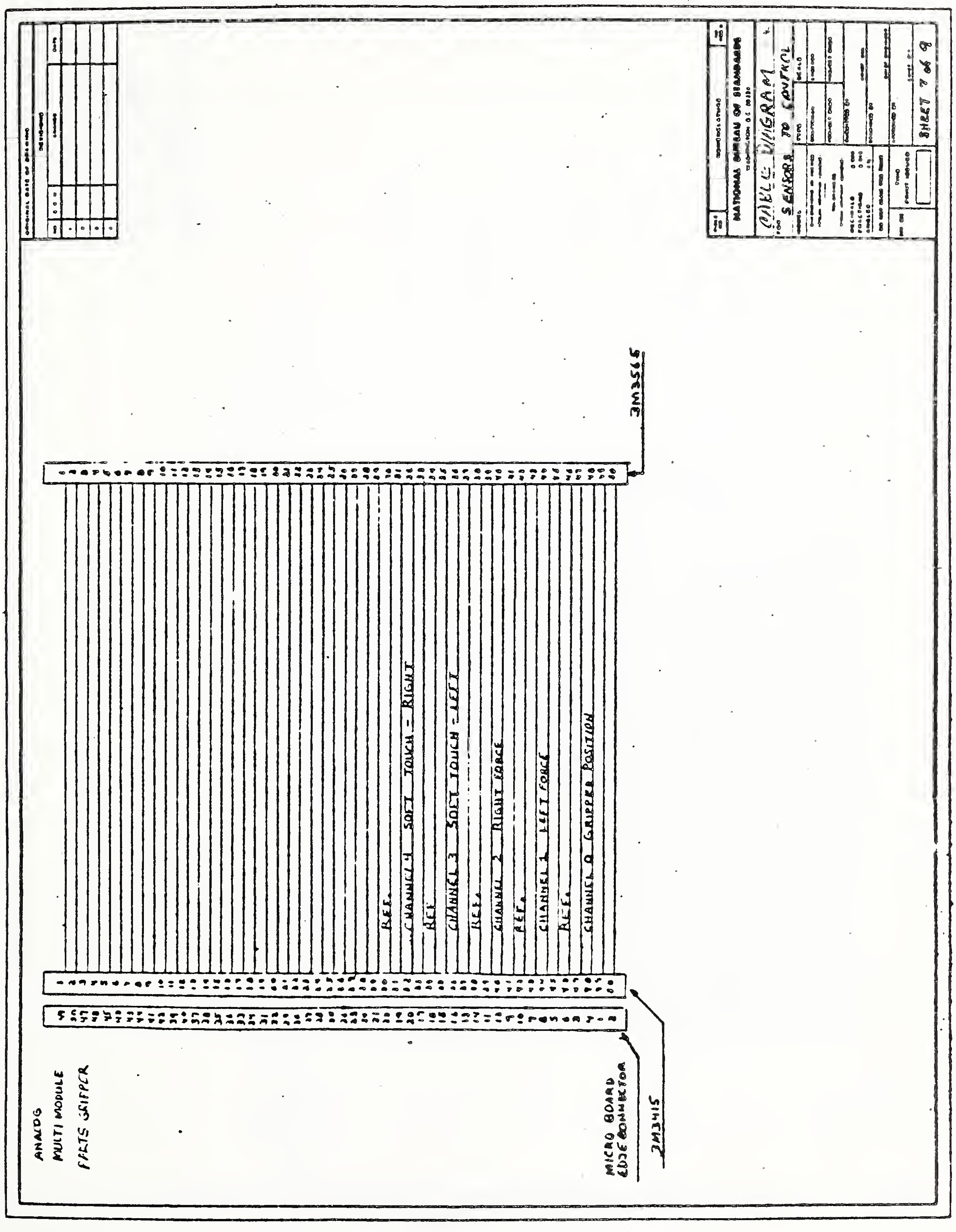

JONES TYPE
BNC
STRAIN GAGE SIGNAL CONDITIONER OUTPUTS
4EA. J32, 33, 34 & 35
ANALOG SIGNALS 7EA.
J25, 26, 27, 28, 29, 30 & 31
SWITCH #1
GND.
ALL SHIELDS
GND.
SIGNAL #1 IN
15V REF.
SIGNAL #2 IN
ACTIVE PEDESTAL JOYSTICK J 17
NATIONAL BUREAU OF STANDARDS
CABLEING DIAGRAM
T3
SHEET 3 of 9

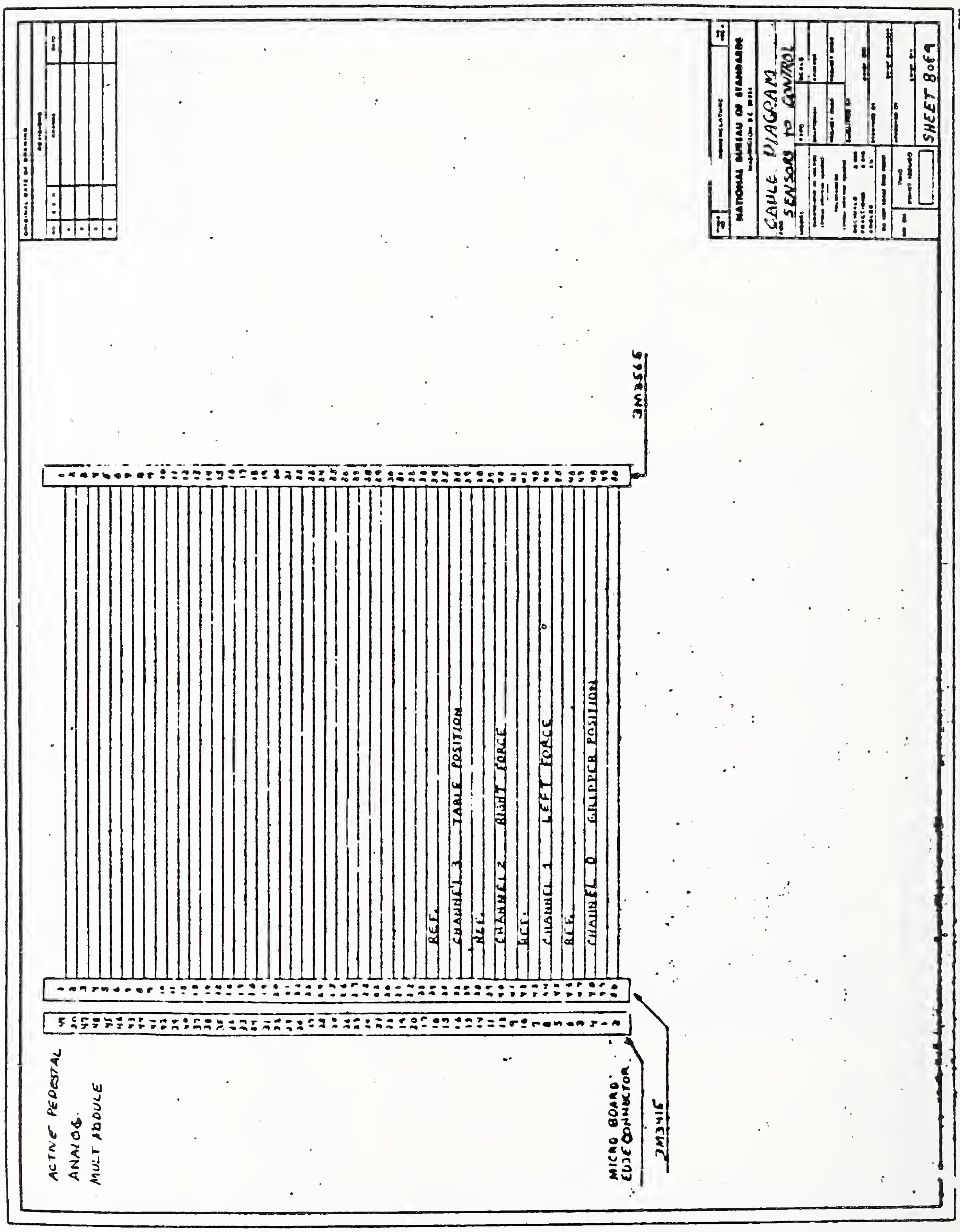

BOARD
EDGE CONNECTOR
BELDEN VARI-TWIST
3M3565
PORT A BIT 0
ROTAC UNLOCK
SOFT TOUCH RIGHT EMITTER
ENABLE MAC VALVES
SERVO VALVE
NATIONAL BUREAU OF STANDARDS
CABLE DIAGRAM
CONTROL TO INTERFACE

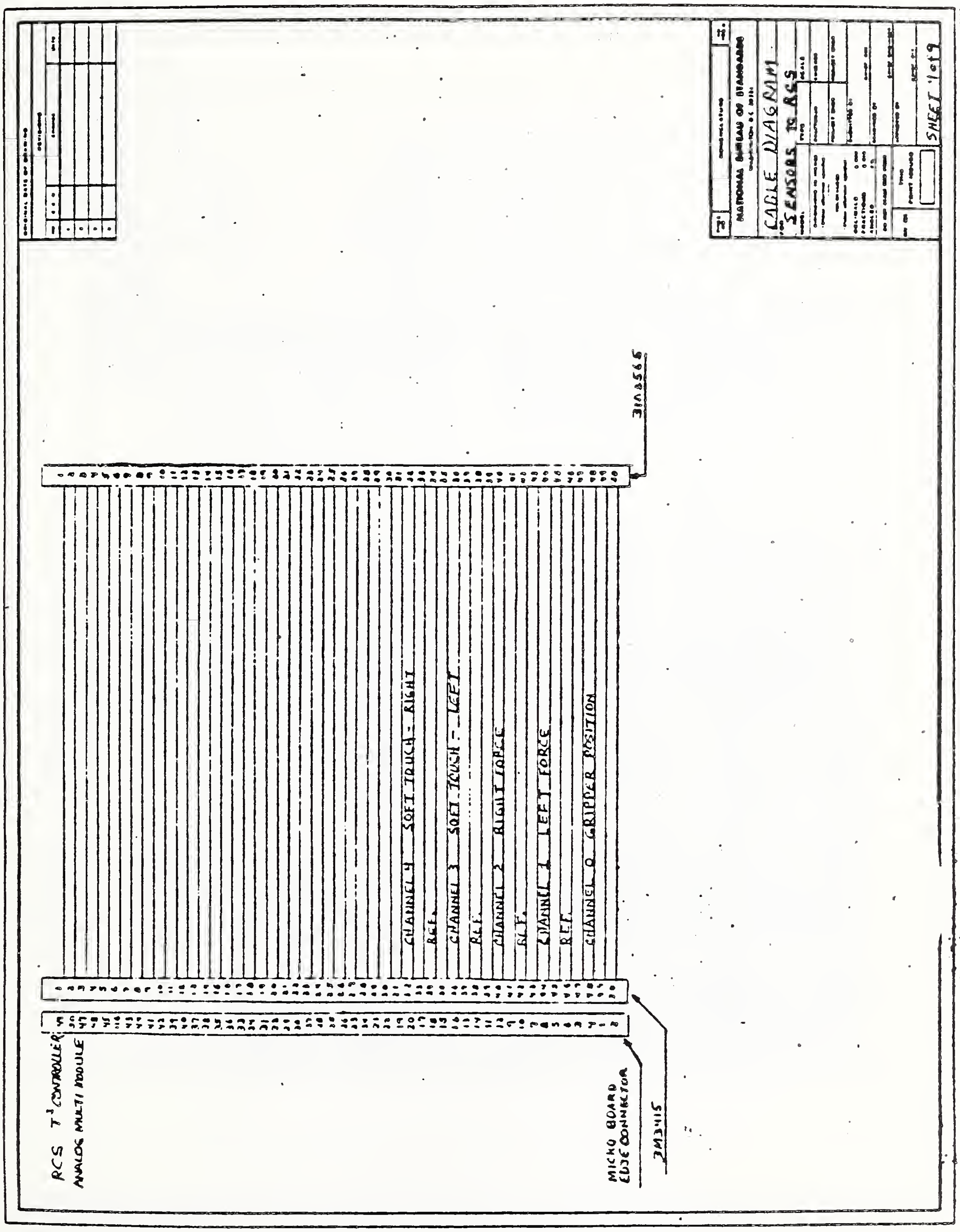

NATIONAL BUREAU OF STANDARDS
CABLE DIAGRAM
CONTROL TO INTERFACE
SHEET 5 of 9
PORT A BIT 0 ACTIVE PEDESTAL
PORT C BIT 3 ACTIVE PEDESTAL
PORT B BIT 0 ACTIVE PEDESTAL
3M3565
3M3415
MICRO BOARD EDGE CONNECTOR



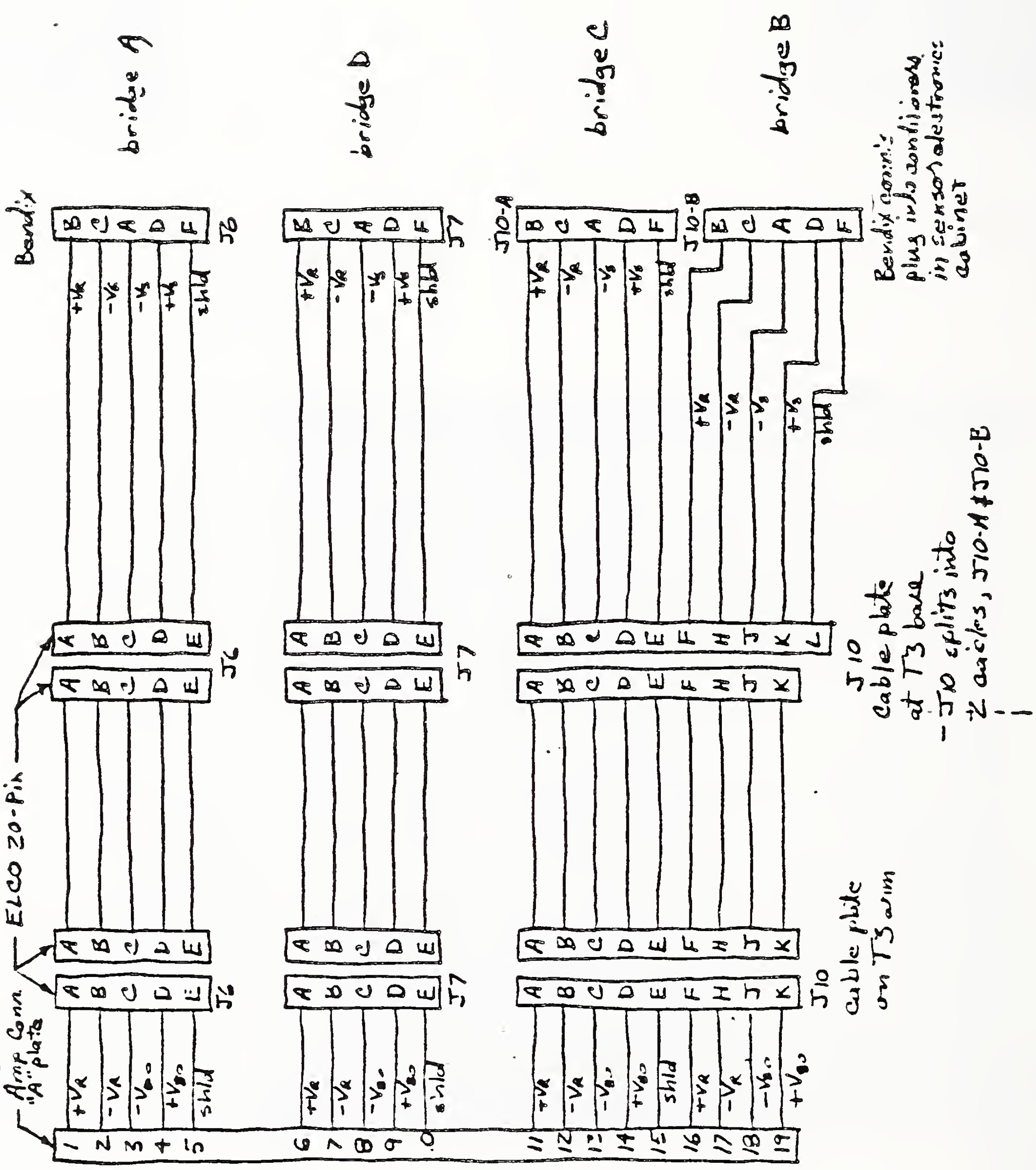
NATIONAL BUREAU OF STANDARDS
CABLE DIAGRAM
MICRO BOARD EDGE CONNECTOR
3M3415

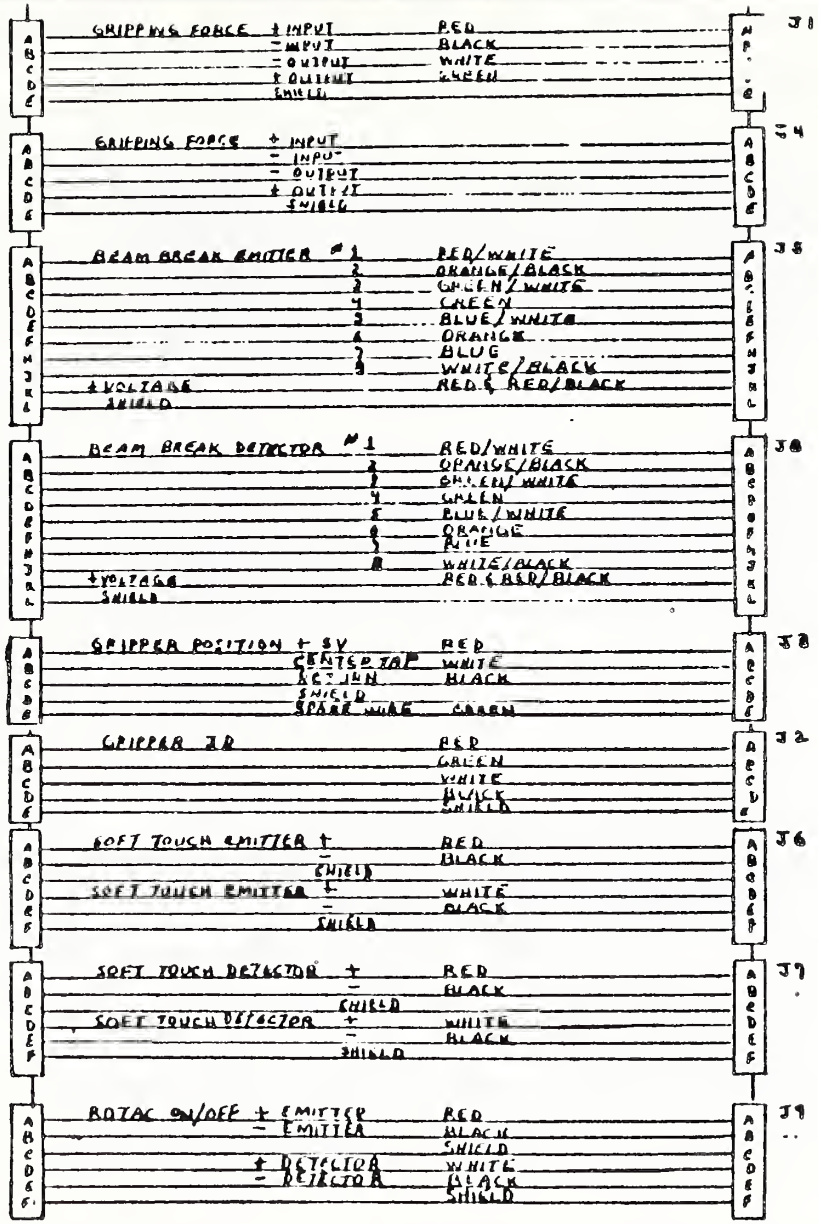

ANALOG
MULTI MODULE
MICRO BOARD EDGE CONNECTOR
3M3415
3M3565
REF.
CHANNEL 4 SOFT TOUCH - RIGHT
REF
CHANNEL 3 SOFT TOUCH - LEFT
REF.
CHANNEL 2 RIGHT FORCE
REF.
CHANNEL 1 LEFT FORCE
REF.
CHANNEL 0 GRIPPER POSITION
NATIONAL BUREAU OF STANDARDS
CABLE DIAGRAM
SENSORS TO CONTROL
SHEET 7 of 9


ACTIVE PEDESTAL
ANALOG
MULT MODULE
MICRO BOARD
EDGE CONNECTOR
3M3415
3M3565
REF.
CHANNEL 3 TABLE POSITION
REF.
CHANNEL 2 RIGHT FORCE
REF.
CHANNEL 1 LEFT FORCE
REF.
CHANNEL 0 GRIPPER POSITION
NATIONAL BUREAU OF STANDARDS
CABLE DIAGRAM
SENSORS TO CONTROL
SHEET 8 of 9



NATIONAL BUREAU OF STANDARDS
CABLE DIAGRAM
SENSORS TO RCS
SHEET 1 of 9
RCS T3 CONTROLLER
ANALOG MULTI MODULE
MICRO BOARD
EDGE CONNECTOR
3M3415
3M3565
CHANNEL 4 SOFT TOUCH - RIGHT
REF.
CHANNEL 3 SOFT TOUCH - LEFT
REF.
CHANNEL 2 RIGHT FORCE
REF.
CHANNEL 1 LEFT FORCE
REF.
CHANNEL 0 GRIPPER POSITION


QC Amp Conn "A" plate
ELCO 20-Pin
J6
J7
J10
cable plate on T3 arm
J10 cable plate at T3 base
- J10 splits into 2 cables, J10-A & J10-B
Bendix
J10-A
J10-B
bridge A
bridge D
bridge C
bridge B
Bendix conn's plug into conditioners in sensor electronics cabinet



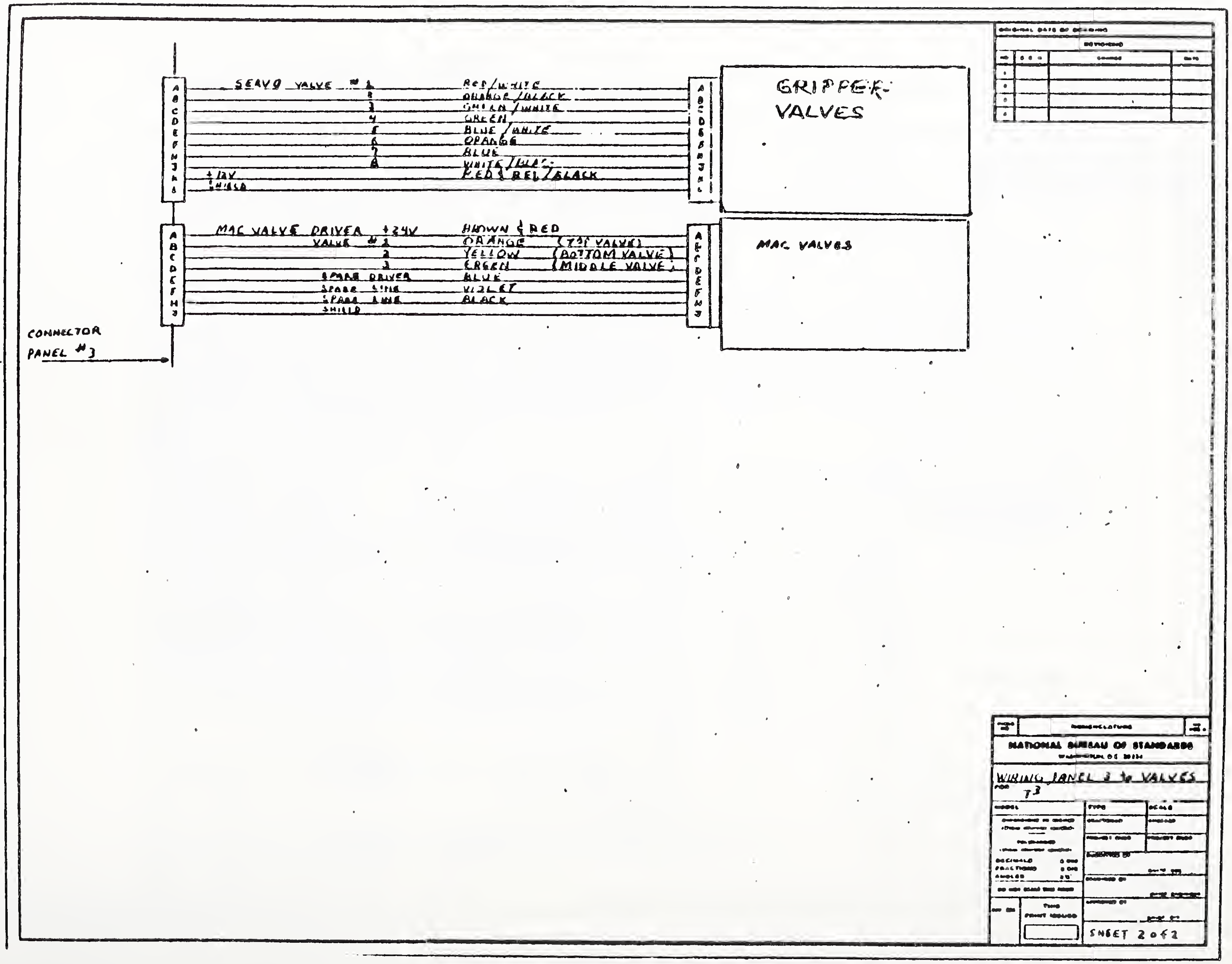
RECEPTACLE KEYING		
J NO.	TOP	BOTTOM
1	1	1
2	1	2
3	1	3
4	1	4
5	1	5
6	1	6
7	2	1
8	2	2
9	2	3

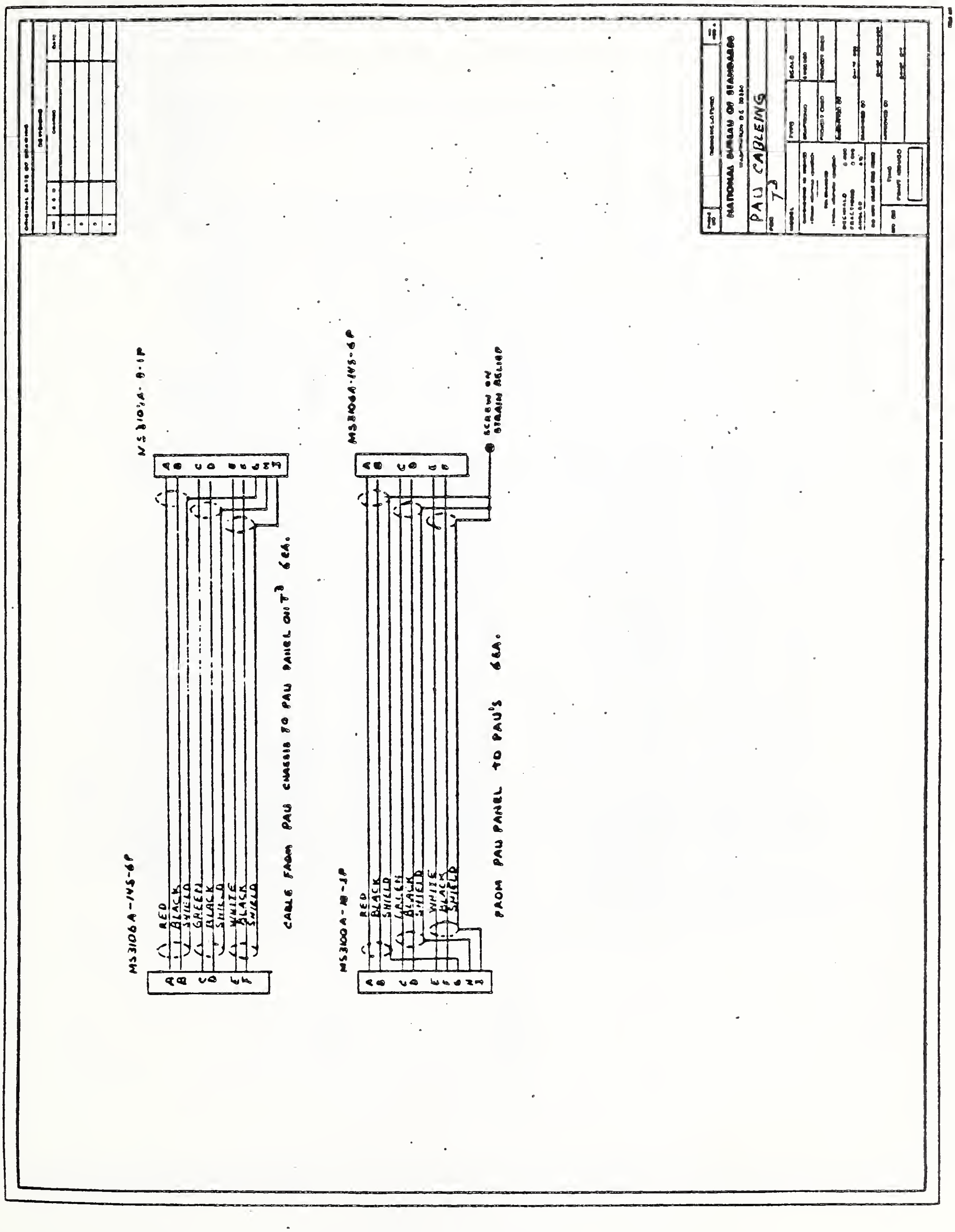
ALL RECEPTACLES
ELCO # 8016-020-000-903

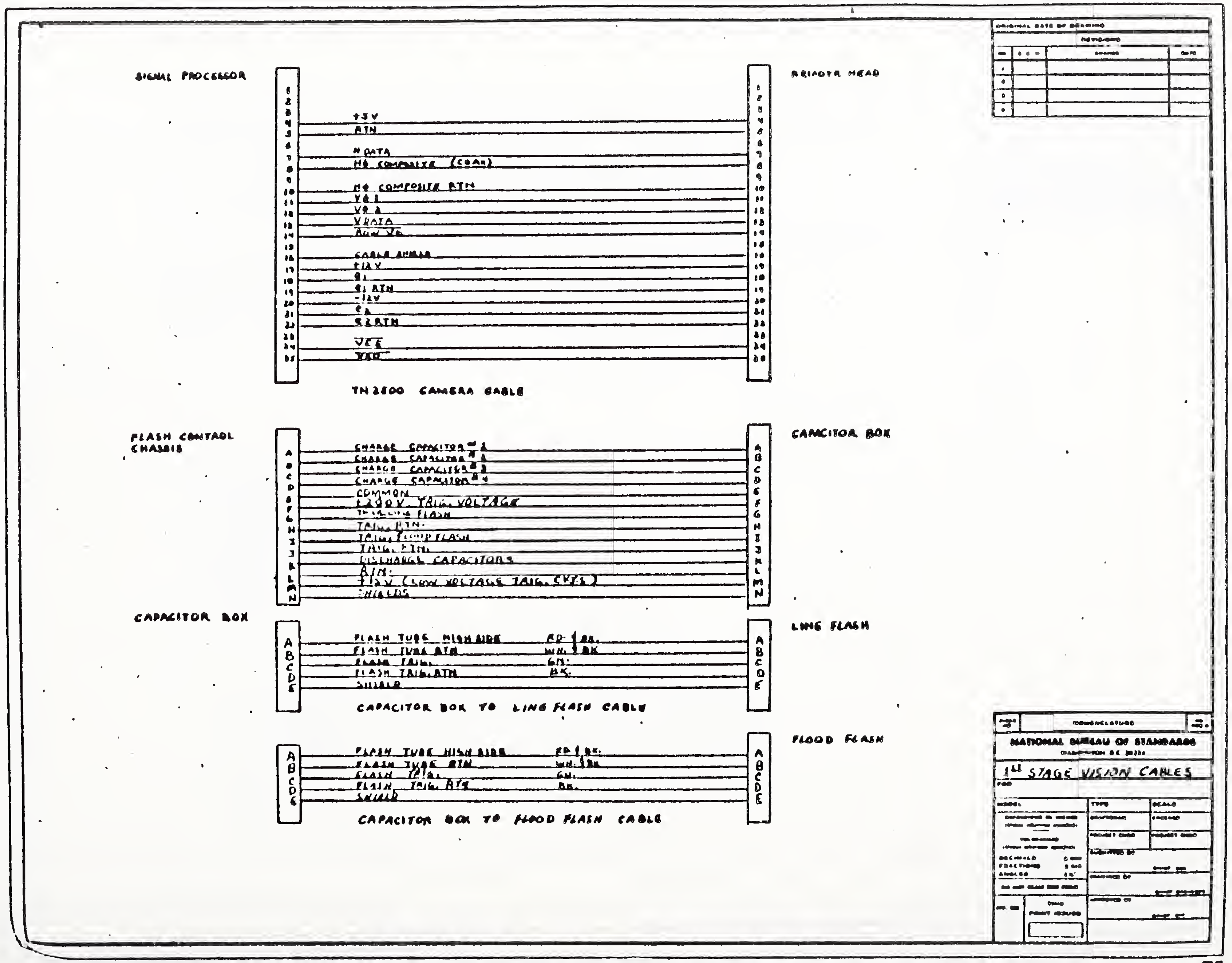
NATIONAL BUREAU OF STANDARDS

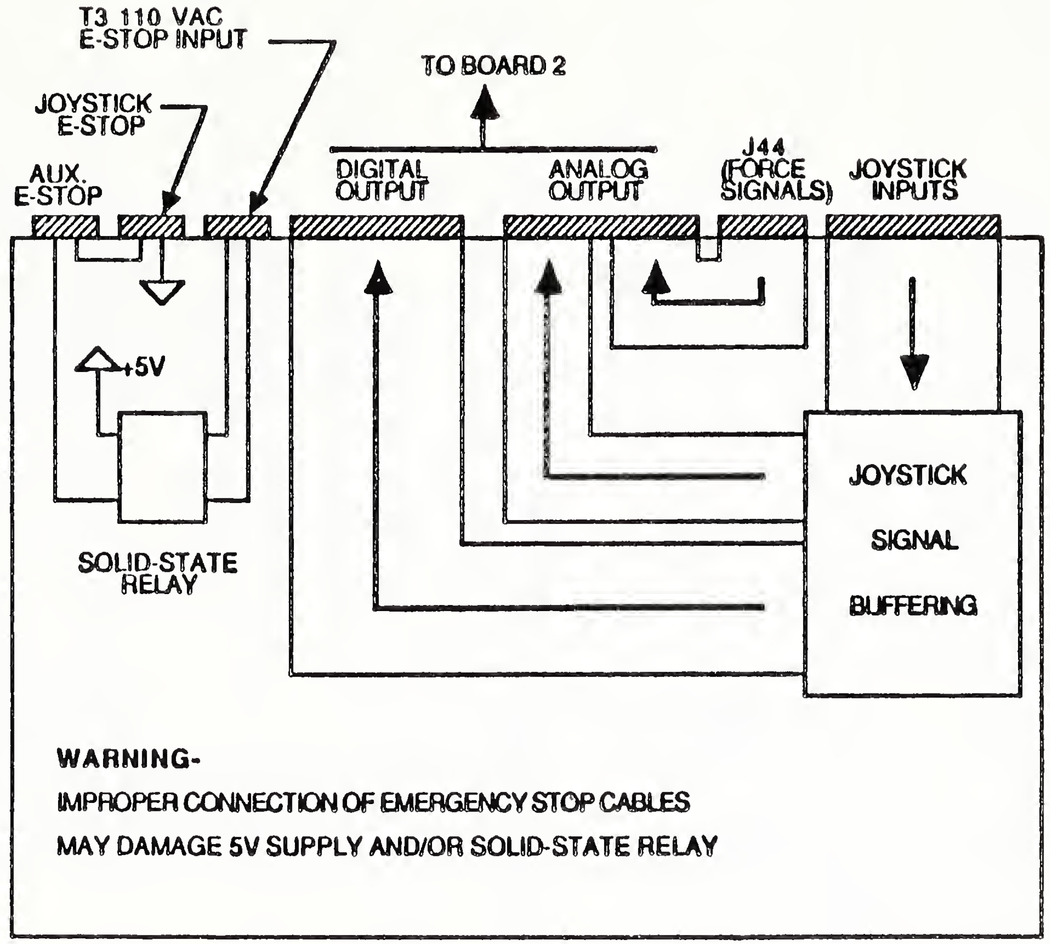
WIRING PANEL, 3 to 2

SHEET 1 OF 1


SERVO VALVE #1
RED/WHITE
ORANGE/BLACK
GREEN/WHITE
GREEN
BLUE/WHITE
ORANGE
BLUE
+12V
SHIELD
RED & RED/BLACK
GRIPPER VALVES
MAC VALVE DRIVER +24V
VALVE #1
SPARE DRIVER
SPARE LINE
SHIELD
BROWN & RED
ORANGE
YELLOW (BOTTOM VALVE)
GREEN (MIDDLE VALVE)
BLUE
VIOLET
BLACK
MAC VALVES
CONNECTOR PANEL #3
NATIONAL BUREAU OF STANDARDS
WIRING PANEL 3 to VALVES
SHEET 2 of 2

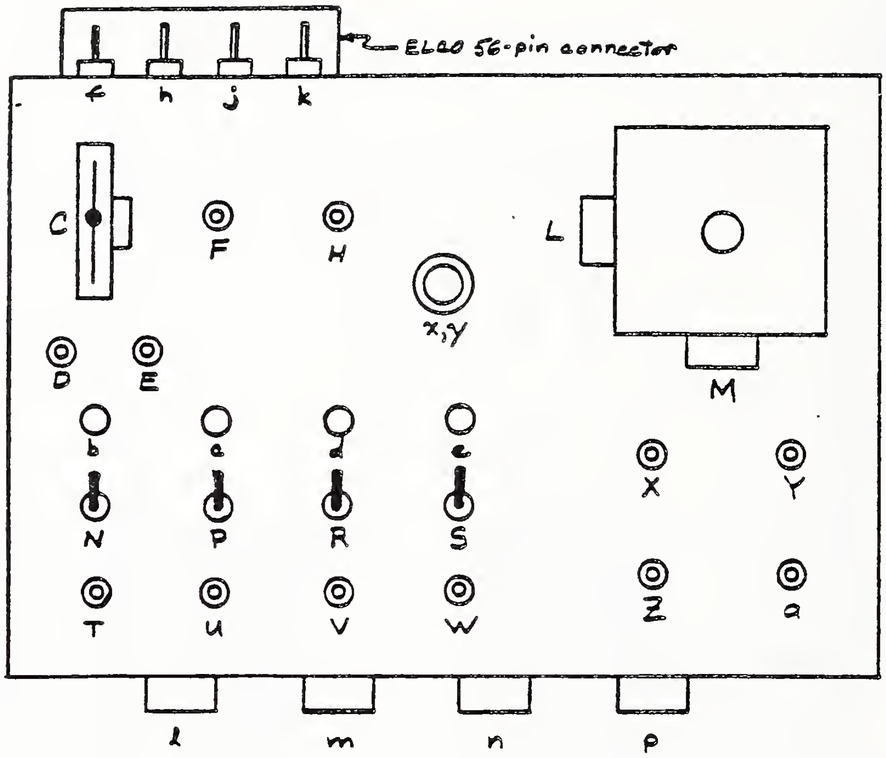

MS3106A-14S-6P
MS3106A-8-1P
RED
BLACK
SHIELD
GREEN
BLACK
SHIELD
WHITE
BLACK
SHIELD
CABLE FROM PAU CHASSIS TO PAU PANEL ON T3 6EA.
MS3100A-18-1P
MS3106A-14S-6P
RED
BLACK
SHIELD
GREEN
BLACK
SHIELD
WHITE
BLACK
SHIELD
SCREW ON STRAIN RELIEF
FROM PAU PANEL TO PAU'S 6EA.
NATIONAL BUREAU OF STANDARDS
PAU CABLEING
T3


SIGNAL PROCESSOR
+5V
RTN
H DATA
COMMON
TN 2500 CAMERA CABLE
FLASH CONTROL CHASSIS
CAPACITOR BOX
LINE FLASH
FLOOD FLASH
FLASH TUBE HIGH SIDE
SHIELD
CAPACITOR BOX TO LINE FLASH CABLE
CAPACITOR BOX TO FLOOD FLASH CABLE
NATIONAL BUREAU OF STANDARDS
1st STAGE VISION CABLES



PRIM INTERFACE BOX

T3 JoysTick (not to scale)



ELCO pin	Function	ELCO pin	Function
A	+5V	Y	extra
B	Gnd.	Z	+roll
C	vel. pot	a	-roll
D	up	b	world coord. LED
E	down	c	Tool coord. LED
F	grasp	d	Tool org. " "
H	release	e	wrisT org. " "
J	N/C	f	world coord. sw.
K	N/C	h	tool coord. sw.
L	fwd/back pot	j	Tool org. coord. sw.
M	left/right pot	k	wrisT org. coord. sw.
N	TASK single-step sw.	l	analog vel (disabled)
P	SUBTASK " " "	m	analog acc. (disabled)
R	E-MOVE " " "	n	pot, not used (disabled)
S	PRIM " " "	p	pot, not used (disabled)
T	TASK " " pb	x	Emerg. stop
U	SUBTASK " " "	y	Emerg. stop
V	E-MOVE " " "		
W	PRIM " " "		
X	RECORD-PT		

LIST OF REFERENCES

[1] The NBS Real-time Control System User's Reference Manual, National Bureau of Standards, to be published.

[2] Simpson, J. A., Hocken, R. J., and Albus, J. S., "The Automated Manufacturing Research Facility of the National Bureau of Standards," Journal of Manufacturing Systems, Vol. 1, No. 1, 1982.

[3] Fishman, D., Scott, H., and Bunch, B., "Integration of Material Buffering Devices in an Automated Factory," Second Int. Conf. on Robotics and Factories of the Future, San Diego, Cal., July, 1987.

[4] Scott, H., and Strouse, K., "Workstation Control in a Computer Integrated Manufacturing System," Proceedings Autofact 6 Conference, Anaheim, CA., October, 1984.

[5] Slocum, A. H., Peris, J., Donmez, A., "Development of a Flexible Automated Fixturing System," SME Conference on Flexible Manufacturing Systems, SME Technical Paper #MR 86-128, February, 1986.

[6] Bunch, R. W., and Vranish, J. M., "'Split Rail' Parallel Gripper," Robots 9 Conference Proceedings, Detroit, June, 1985.

[7] Vranish, J. M., "'Quick Change' System for Robots," Robots 8 Conference Proceedings, Detroit, June, 1984.

[8] J. Albus, E. Kent, M. Nashman, P. Mansbach, L. Palombo, M. O. Shneier, "Six Dimensional Vision System," SPIE, Vol 336, Robot Vision, 1982, p. 142.

[9] M. O. Shneier, R. Lumia, E. W. Kent, "Model Based Strategies for High-Level Robot Vision," CVGIP, Vol. 33, 1986, p. 293.

[10] Barbera, A. J., Fitzgerald, M. L., Albus, J. S., and Haynes, L. S., "RCS: The NBS Real-time Control System," Robots 8 Conference Proceedings, Detroit, June, 1984.

[11] Fitzgerald, M. L., Barbera, A. J., and Albus, J. S., "Real-time Control Systems for Robots," Proceedings 1985 SPI National Plastics Exposition Conference, Chicago, June, 1985.

[12] Fiala, J. C., Wavering, A. J., "An RCS Application Example:

Tool Changing on a Horizontal Machining Center," Second Int. Conf. on Robotics and Factories of the Future, San Diego, Cal., July, 1987.

[13] 2100 System Strain Gage Conditioner and Amplifier System Instruction Manual, Vishay Instruments, Raleigh, N. C., 1977.

[14] Operating Manual for Cincinnati Milacron T3 Industrial Robot, Publication No. 1-IR-79149, Cincinnati Milacron Co., Cincinnati, Ohio, 1980.

NBS-114A (REV. 2-8C)

U.S. DEPT. OF COMM. **BIBLIOGRAPHIC DATA SHEET** *(See instructions)*	1. PUBLICATION OR REPORT NO. NBSIR 88-3692	2. Performing Organ. Report No.	3. Publication Date FEBRUARY 1988

4. TITLE AND SUBTITLE

The Real-Time Control System of the Horizontal Workstation Robot

5. AUTHOR(S)

Albert J. Wavering, John C. Fiala

6. PERFORMING ORGANIZATION *(If joint or other than NBS, see instructions)*

NATIONAL BUREAU OF STANDARDS
DEPARTMENT OF COMMERCE
WASHINGTON, D.C. 20234

7. Contract/Grant No.

8. Type of Report & Period Covered

9. SPONSORING ORGANIZATION NAME AND COMPLETE ADDRESS *(Street, City, State, ZIP)*

10. SUPPLEMENTARY NOTES

☐ Document describes a computer program; SF-185, FIPS Software Summary, is attached.

11. ABSTRACT *(A 200-word or less factual summary of most significant information. If document includes a significant bibliography or literature survey, mention it here)*

This manual describes the real-time control system used to control the robot in the Horizontal Workstation of the Automated Manufacturing Research Facility.

12. KEY WORDS *(Six to twelve entries; alphabetical order; capitalize only proper names; and separate key words by semicolons)*

13. AVAILABILITY

☒ Unlimited
☐ For Official Distribution. Do Not Release to NTIS
☐ Order From Superintendent of Documents, U.S. Government Printing Office, Washington, D.C. 20402.
☒ Order From National Technical Information Service (NTIS), Springfield, VA. 22161

14. NO. OF PRINTED PAGES
183

15. Price
$18.95

USCOMM-DC 6043-P80